UNIVERSITY OF CALGARY

Quantum Walk Schemes

for Universal Quantum Computation

by

Michael S. Underwood

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF PHYSICS AND ASTRONOMY

CALGARY, ALBERTA

January, 2013

Abstract

Random walks are a powerful tool for the efficient implementation of algorithms in classical computation. Their quantum-mechanical analogues, called quantum walks, hold similar promise. Quantum walks provide a model of quantum computation that has recently been shown to be equivalent in power to the standard circuit model. As in the classical case, quantum walks take place on graphs and can undergo discrete or continuous evolution, though quantum evolution is unitary and therefore deterministic until a measurement is made. This thesis considers the usefulness of continuous-time quantum walks to quantum computation from the perspectives of both their fundamental power under various formulations, and their applicability in practical experiments.

In one extant scheme, logical gates are effected by scattering processes. The results of an exhaustive search for single-qubit operations in this model are presented. It is shown that the number of distinct operations increases exponentially with the number of vertices in the scattering graph. A catalogue of all graphs on up to nine vertices that implement single-qubit unitaries at a specific set of momenta is included in an appendix. I develop a novel scheme for universal quantum computation called the discontinuous quantum walk, in which a continuous-time quantum walker takes discrete steps of evolution via perfect quantum state transfer through small 'widget' graphs. The discontinuous quantum-walk scheme requires an exponentially sized graph, as do prior discrete and continuous schemes.

To eliminate the inefficient vertex resource requirement, a computation scheme based on multiple discontinuous walkers is presented. In this model, n interacting walkers inhabiting a graph with $2n$ vertices can implement an arbitrary quantum computation on an input of length n, an exponential savings over previous universal quantum walk schemes. This is the first quantum walk scheme that allows for the application of quantum error correction.

The many-particle quantum walk can be viewed as a single quantum walk undergoing perfect state transfer on a larger weighted graph, obtained via equitable partitioning. I extend this formalism to non-simple graphs. Examples of the application of equitable partitioning to the analysis of quantum walks and many-particle quantum systems are discussed.

Acknowledgements

It is a pleasure to be able to thank my supervisor, Dr. David Feder, for the experiences I have gained while working with him. His advice and guidance have been invaluable, and his enthusiasm for a wide range of fields academic and otherwise has made my transition from first-year student to a senior member of our research group more enjoyable than I've been led to believe grad school has a right to be. That journey would not have been the same had it not been shared with my friend and colleague, Adam D'Souza. I will miss the sense of camaraderie and invigorating inquisitiveness evoked at our weekly group meetings.

I am grateful to Drs. Barry Sanders and Peter Høyer for their thoughtful guidance and helpful comments throughout the course of this research. Our discussions have done much to shape the direction of my work for the better. I acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada, and from Alberta Innovates Technology Futures.

The support and encouragement I have received from my family—my grandfather, brother and sister-in-law, my parents-in-law, and of course my parents—while I pursued this goal have been immeasurable. Thank you all. The students and staff who have shaped my time here are too many to name, but in particular I would like to acknowledge Jérémie Choquette, Michael Cummings, Mahdi Ebrahimi Kahou, Timothy Friesen, Jacob Foster, Patrick Irwin, Itzel Lucio Martinez, Farokh Mivehvar, Brian Niebergal, Julia Pulwicki, and Michael Skotiniotis.

Above all I cannot express the magnitude of the support and inspiration I receive from my wife Katie each and every day. This thesis is dedicated to her.

For Katie

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
A, $A(G)$, A_G	An adjacency matrix (of the graph G)
B	The adjacency matrix for a collapsed graph $A(G/\Pi)$
$\mathbb{C}$	The complex numbers
$c^\dagger$, c	(Bosonic) walker creation and annihilation operators
δ_{ij}	The Kronecker delta, equal to 1 if $i = j$, 0 otherwise
$\doteq$	A defining equality, or one that holds by definition
$\stackrel{!}{=}$	Required equality that may not hold generally
g	On-site interaction strength
G	A graph, containing an edge set E and vertex set V
$\mathcal{G}$	A gate set
H	Hadamard matrix
$\mathcal{H}$	Hamiltonian
H.c.	Hermitian conjugate
$\mathscr{H}$	Hilbert space
I_n	The $n \times n$ identity matrix
K_n	The complete graph on n vertices
μ_v	A local potential or self-loop applied to vertex v
$\mathcal{N}$	The number operator $c^\dagger c$
$O(\cdot)$	Big O; roughly, 'at most $\cdot$'
$\Omega(\cdot)$	Big Omega; roughly, 'at least $\cdot$'
$\varnothing$	The empty set
P	A partition matrix
P_n	The path graph on n vertices

Π	A partition
Q	A normalized partition matrix
$\mathbb{Q}$	The rational numbers
$R_l(\theta)$	A rotation of the Bloch sphere about the axis l by angle θ
$\mathbb{R}$	The real numbers
τ	Hopping or transition rate under the Bose–Hubbard model
tilde, $\tilde{v}$	With respect to a partition, the index of the cell containing vertex v
$\Theta(\cdot)$	Big Theta; roughly, 'the same as $\cdot$'
U	A unitary operator
V	A set of vertices
X, Y, Z	The Pauli matrices
$\mathbb{Z}$, $\mathbb{Z}_n$	The integers, integers modulo n

Chapter 1

Universal computation

1.1 Classical computation

Classical computation, as can be performed on an abacus, supercomputing cluster, or anything in between, is at its most basic level nothing but the storage and manipulation of information. The fundamental unit of information is the bit, and a single bit can take on either of a pair of mutually exclusive values—True or False, yes or no, +5 V or GND—which are generally abstracted to be 0 and 1. The goal of computing is to solve problems, or answer questions. Where mechanical and electronic devices provide a distinct advantage in attaining this goal is in cases for which the same question may be asked for a wide range of parameters, and a set of steps to find the answer can be laid out in advance that will produce the correct answer regardless of which values are supplied to the parameters. The set of steps is an *algorithm*, the values supplied for a given instance of the parameters are the *input* to the algorithm, and the resulting answer is its *output*. For example, the Pythagorean theorem allows the length c of the hypotenuse of a right triangle to be found by taking the square root of the sum of the squares of the lengths of the other two sides, a and b. This is true regardless of the values of a and b that are provided, so long as they are positive real numbers. In this example the input is the pair of values $\{a, b\}$, the algorithm is defined by the function

$$\begin{array}{rccl} f_c: & (\mathbb{R}^+, \mathbb{R}^+) & \to & \mathbb{R}^+ \\ & (a, b) & \mapsto & c = \sqrt{a^2 + b^2}, \end{array} \tag{1.1}$$

and the output is the value of c.

Much as, for example, the properties of the real numbers are not studied by attempt-

ing to investigate each member of the continuum, a discussion of computation cannot proceed solely by looking at individual algorithms. For this reason, a formalism dating back to Gödel [1], Church [2], and Turing [3] has been developed to abstract the fundamental mathematical aspects constituting a computation away from the physical processes involved in any particular implementation thereof.

1.1.1 Turing machines

What Turing proposed was a hypothetical machine, which now bears his name, that embodies a specific *model* of computation. The detailed workings of a Turing machine are unimportant here, save for one distinction between two classes of the machines. Deterministic Turing machines are such that given a fixed input, they will always produce the same output (if any), and in the same sequence of operations. The set of operations available to a deterministic Turing machine consists only of functions in the mathematical sense. On the other hand a non-deterministic Turing machine can make random choices during its operation, so that a given input to an operation may lead to one of many outputs. Therefore having a non-deterministic Turing machine execute an algorithm repeatedly on a fixed input can lead to different outputs, and different times taken to arrive at those outputs. There are multiple ways in which such randomness can be employed. A *randomized* or *non-deterministic algorithm* always produces a correct result in a random but bounded time if one exists (or correctly identifies in bounded time that no such result exists), however if multiple correct results exist then which of them is returned is a random variable. *Probabilistic algorithms* on the other hand have a known probability to either produce an incorrect result, or to fail to produce a result. Algorithms of the first kind are referred to as 'Monte Carlo algorithms', while those of the second kind are 'Las Vegas algorithms'.

Remarkably, Turing was able to show that in the deterministic case there exists a

universal Turing machine, in that one can describe a single Turing machine capable of simulating any other, and therefore of computing any function that is computable by Turing machines. Independently Church developed a distinct model of computation known as the λ-calculus, and Turing showed in an appendix to Reference [3] that a function is calculable under the λ-calculus if and only if it is computable by a Turing machine. That is, he proved that the two models of computation are equivalent, meaning that the λ-calculus is *Turing complete*. Furthermore the non-deterministic Turing machine adds nothing to the class of functions that can be computed by machine. In fact, it has come to be accepted that the rigorously defined Turing-machine model of computation encapsulates the intuitive concept of what it means to say that a function is algorithmically computable. This is presented more formally by the Church–Turing thesis, which can be stated as follows:

> *The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.* *(Reference [4], p. 125)*

1.1.2 Algorithms and computational problems

The Church–Turing thesis has led to the definition of a *computable function* as a function such that there exists a Turing machine that computes it. A related concept is that of *decision problems*, in which a specified predicate is to be found either true or false, dependent upon the input. For example, if x and $k < x$ are positive integers, then the predicate 'x has a prime factor less than or equal to k' is either true or false, and determining which is the case is a computational decision problem.

Note that whether a function is computable (or predicate decidable) and whether this can be done efficiently are two distinct issues. Furthermore, there exist randomized algorithms that can solve problems significantly more quickly on average than is possi-

ble with only deterministic operations. For example, no deterministic polynomial-time algorithm can approximate the volume of an arbitrary convex subset of d-dimensional Euclidean space to within a multiplicative factor that is exponential in d [5]. That is, any deterministic algorithm to perform this approximation to arbitrary accuracy must take at least an exponential amount of time. Nevertheless, there exists a randomised algorithm that runs in polynomial time and yet can provide such an arbitrarily accurate approximation [6]. While the volume can be approximated with or without randomness, making use of it provides an exponential speed-up to the running time.

The concept of universal computation allows algorithms to be studied in their own right, independently of the computer chosen to run them, by abstracting computation under a model of computing away from any specific physical architecture that might implement the model. The result is the field of *computational complexity*, in which the focus is on the behaviour of an algorithm in terms of how long it takes to run and how much memory it requires, as functions of the size of its input. For instance, it is known that as the size N of a list grows toward infinity, sorting the list takes on average a number of operations proportional to $N\log(N)$, to first order. Therefore doubling the length of the list roughly doubles the time required to sort it. This comparison of relative times remains the same whether the computer doing the sorting is the latest supercomputer, a programmable calculator, or ENIAC,[1] even though the absolute times taken to sort any specific list will vary drastically among these different devices.

Decision problems that accept an input of length N and can be decided in a time that is a polynomial function of N belong to the *complexity class* P. More specifically, P is the class of all decision problems that can be solved by a deterministic Turing machine in a time that is bounded above by a polynomial function of the input size. For the purposes of computational complexity, a process is said to be *efficient* with respect to a

[1] Unveiled in 1946, the Electronic Numerical Integrator And Computer was the world's first computationally universal electronic computer.

resource such as time whenever its use of that resource is polynomial. Thus P consists of all decision problems that can be solved efficiently by deterministic Turing machines. Another important complexity class is known as NP, for 'non-deterministic polynomial'. One can think of NP as the class of decision problems for which it is efficient to verify with a deterministic Turing machine that a proposed solution is correct, but for which it is not necessarily efficient to find correct solutions. One such problem has already been mentioned: Given positive integers x and $k < x$, does x have a prime factor $p \leq k$? If such a prime number is proposed, determining whether it divides x is easily achieved in polynomial time. However, there is no known classical algorithm that is capable of producing such a p for arbitrary x and k in a time polynomial in the number of digits in each of x and k. Note however that it is not known that an efficient classical algorithm is impossible, so it may turn out that this problem is in P. In fact, it is not even known whether there exist any problems in P that are not in NP! That is, while it is clear that $\mathrm{P} \subseteq \mathrm{NP}$, it is unknown whether or not $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$. The generalizations of P and NP to computational (or functional) problems, such as for example finding all prime factors of an integer x, are known as FP and FNP, respectively.

Along with studying individual algorithms without needing to appeal to the specific computer architecture that may implement them, another advantage of the abstraction of computation is the ability to consider multiple models of computation, and compare them all on an equal footing. Given a proposal for a new model of computing, one needs only to show that it can simulate an arbitrary Turing machine, and conversely that a Turing machine can simulate the new model, in order to know that the model is universal and equal in power to all previous models of universal computation. It will be true of course that the new model provides no new power in terms of which functions can or cannot be computed, since it is equivalent to the Turing machine model. But just as it is useful to be able to have multiple methods for solving specific problems—say, finding

the length of a hypotenuse from Equation (1.1), or by using trigonometry, or simply with a ruler—new models can provide new ways of thinking about problems. Often a certain model is particularly well suited to the description of a certain problem, motivating the development of new classes of algorithms that can be implemented in any model, but are perhaps only intuitive in one. One particular model of computation that carries over particularly well to the case of quantum computing is the so-called circuit model of computation.

1.1.3 The circuit model

As illustrated by the example of computing f_c in Equation (1.1), a computation is a process that accepts a set of input parameters, performs a calculation based on them, and returns an output. In the case of f_c the input parameters a and b and the resulting output c can be arbitrary positive real numbers. Any real computer must be finite however, and therefore can only make use of a finite number of bits when encoding information. The *circuit model* demonstrates this fact explicitly, while also being Turing complete. A *circuit* is composed of wires that carry bits of information, and gates that enact transformations upon those bits. A logic gate on k bits performs a function $f : \{0,1\}^k \to \{0,1\}^l$ for some positive integer l that may or may not be equal to k. For example the NOT gate acts on a single bit, transforming 0 to 1 and 1 to 0; the exclusive-or gate XOR accepts two input bits and outputs a single bit equal to 1 if exactly one of the inputs is 1, otherwise it outputs 0; and the three-bit Toffoli gate does nothing to two of its input bits, but performs a NOT on the third bit only if the first two are both equal to 1. For this reason the Toffoli gate is also called the controlled-controlled-NOT or C-CNOT gate, since the values of the first two bits control whether a NOT operation is applied to the third. Figure 1.1 shows an example three-bit circuit containing these three gates and defining an example logical function $f : \{0,1\}^3 \to \{0,1\}^2$. The input bit string $a_2a_1a_0$ is

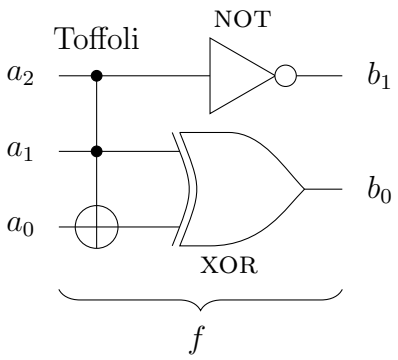


Figure 1.1: Example classical circuit mapping three input bits to two outputs.

mapped to the output $b_1 b_0$ according to the following truth table:

$$\begin{array}{lccc} f: & \{0,1\}^3 & \to & \{0,1\}^2 \\ & 000 & \mapsto & 10 \\ & 001 & \mapsto & 11 \\ & 010 & \mapsto & 11 \\ & 011 & \mapsto & 10 \\ & 100 & \mapsto & 00 \\ & 101 & \mapsto & 01 \\ & 110 & \mapsto & 00 \\ & 111 & \mapsto & 01. \end{array} \tag{1.2}$$

Note that in Figure 1.1 the NOT and XOR gates appear above each other, indicating that they are to be executed simultaneously. The output bits b_0 and b_1 do not depend on whether the NOT is applied to the first bit before, after, or concurrently with the application of XOR to the second and third bits. These two gates can therefore be *parallelized*. There is however no way to execute either of these gates at the same time as the Toffoli gate in this circuit. That is, there are two distinct *steps* to the circuit, which is therefore able to be drawn as two columns of gates across the wires. This leads to the concept of the *depth* of a circuit, which is the minimum number of steps or columns in which the gates can be performed. The circuit depth of f in this case is two. A related concept is the *width* of a circuit, which is equal to the maximum number of parallel wires appearing throughout its depth; in this case the width of the circuit for f is three.

In general a problem to be solved under the circuit model can reasonably be expected to have inputs of many different lengths. A specific circuit however is of a fixed size. For instance, one might desire an algorithm that accepts two integers x and y, and returns their product xy, regardless of the number of digits in either integer. No single circuit can accomplish this, since however many wires it contains one might be presented with values for x and y requiring a greater number of bits. For this reason, while it is not uncommon to colloquially speak of 'the' circuit for multiplication, what is meant is the *family* of circuits that implement the functions $f_{mn}(x, y) = xy$, where the inputs x and y are m- and n-bit integers, respectively, and m and n are allowed to independently range over all positive integers. That is, there is a fixed finite circuit for each ordered pair (m, n), and the family consists of the infinite set containing all such circuits.

1.1.4 Universal classical circuits

It turns out, rather remarkably, that the set of building blocks required to construct an arbitrary computable function on an arbitrary number of bits contains only a few elements. Along with wires that carry bit values, and which are allowed to cross over each other to swap two bits, one needs only be able to provide ancillary bits initialized in a standard state, say 0, and perform the logic gates from one of several small sets. One such set consists of the two gates NAND and FANOUT. The FANOUT gate makes copies of bits; its defining map is

$$\begin{array}{rccc} \text{FANOUT}: & \{0,1\} & \to & \{0,1\}^2 \\ & 0 & \mapsto & 00 \\ & 1 & \mapsto & 11. \end{array} \tag{1.3}$$

The NAND gate, an abbreviation for NOT-AND, does exactly those operations: it maps two input bits to their logical-AND, and then inverts the result. That is, NAND maps two input bits to 1 unless they are both initially equal to 1 themselves, according to

$$\begin{array}{rccc} \textsc{nand}: & \{0,1\}^2 & \to & \{0,1\} \\ & 00 & \mapsto & 1 \\ & 01 & \mapsto & 1 \\ & 10 & \mapsto & 1 \\ & 11 & \mapsto & 0. \end{array} \tag{1.4}$$

Copies of NAND and FANOUT can be used to construct a circuit that implements any finite logic gate on an arbitrary number of bits. Thus these two gates are said to form a *universal gate set* for classical computation. In discussions of such gate sets it is generally assumed that the wires and ancillary bits are given. In fact FANOUT is usually assumed as well, so that it is common to say for example that the one-gate set {NAND} is universal. Of course, it is certainly not the only universal gate set. For example, with NOT and AND it is trivial to construct a NAND gate, and therefore {AND, NOT} is a second universal set. An important point is that any universal set of gates is not only equivalent to every other in terms of what it can compute, but also in terms of how many copies of the gates in the set are required to implement any given algorithm. 'Equivalent' in this context means that if an algorithm can be constructed from m copies of the gates from one universal set, then it can be built with gates from another universal set using a number of copies that is a polynomial in m. Note also that due to the use of FANOUT and the fact that logic gates such as NAND are many-to-one, it is possible for the width of a circuit to be greater than both the number of input bit and the number of output bits.

1.2 Introduction to quantum computation

In his keynote address at the MIT Physics of Computation conference in 1981, Feynman introduced—in an aside—the concept of "a new kind of computer—a quantum computer", which he described as "not a Turing machine, but a machine of a different kind" [7]. His motivation at the time was the computational simulation of quantum systems,

which even on today's best computers more than thirty years later can only be simulated in complete generality for on the order of a few tens of particles. This is because a system of n particles each of dimension d evolves in a Hilbert space of dimension d^n, and a complex coefficient must be kept track of for each basis state of the space. A standard single-precision real number in most modern computer architectures requires 32 bits to store; two such real numbers are required to encode a complex value. Even for spin-$\frac{1}{2}$ particles with $d = 2$, a fifty-particle system requires over nine petabytes (over 9×10^{15} bytes) of memory simply to store the state at any given time, and that does not take into account the diagonalization of the 2^{50}-dimensional complex-valued Hamiltonian. A few petabytes of memory may not be far beyond the limits of extant supercomputers[2] but the fact that each additional simulated particle further doubles the memory requirements simply makes such simulations infeasible as a general solution. Furthermore, single-precision floating-point numbers do not come close to the *exact* simulation in which Feynman was interested. His questions then were, could such simulations be performed efficiently on a computer "with quantum computer elements", and if so, what elements would comprise a universal quantum computer, and which classes of quantum systems would be intersimulable?

Such simulations are indeed a significant motivating factor for the development of quantum computers, but Feynman's discussion also asked another question that has led to the development of the field of quantum information science, though he paid even less attention it during his aside. If physical systems are inherently quantum mechanical, and Turing machines are distinctly not, might there exist functions that are computable in the intuitive sense—ones for which physically implementable algorithms can be devised—that are nevertheless not computable by Turing machines? That is, might the Church–Turing

[2]For example, at the time of writing the fastest supercomputer in the world is the IBM Sequoia BlueGene/Q installed at Lawrence Livermore National Laboratory in the US, which has 1.57 PB of memory and can perform 16.32×10^{15} floating-point operations per second [8].

thesis fail when quantum computers are considered? To date the answer appears to be 'No' in terms of the set of functions that are computable, but quantum computing remains an active field of both theoretical and experimental research because it has been shown that the time taken to calculate certain computable functions can be significantly decreased if a quantum computer is available.

Motivated by the potential capabilities of quantum computers in the algorithmic, rather than simulatory, sense, Deutsch developed in 1985 a fully quantum model for computation, describing a quantum generalization of the universal Turing machine [9]. The central idea is to initialize a quantum system in a state that encodes a classical bit string as input, and then allow the system to evolve quantum mechanically through states that encode arbitrary superpositions of bit strings. Finally, a measurement is performed that collapses the state of the system such that it once again encodes a single classical bit string; this is the output of the computation. In analogy to the classical case, any model for quantum computation that can both be simulated by a quantum Turing machine and simulate an arbitrary one, is a model for universal quantum computation. With the shift from classical to quantum Turing machines, come additional complexity classes. In particular, the quantum generalizations of P and NP are BQP, for 'bounded-error quantum polynomial-time', and QMA, for 'quantum Merlin-Arthur', respectively. These classes are not defined as simply as by replacing the Turing machines in those of P and NP with their quantum counterparts, however thinking of them as such is a useful heuristic.

In the same paper, Deutsch also provided the first example of a task that a quantum computer could perform in fewer steps than a classical one. Given a function on a single bit, $f : \{0,1\} \to \{0,1\}$, it must be the case that either $f(0) = f(1)$, or $f(0) \neq f(1)$. Classically the only way to determine which of these cases applies is to evaluate both $f(0)$ and $f(1)$, then compare the results. Quantum mechanically however, Deutsch's

algorithm can be used to determine which case holds with only a single call to the function f. This was a remarkable first step for quantum computers, though of little practical application. Furthermore, it turns out that Deutsch's algorithm has four output values: '$f(0) = f(1)$', '$f(0) \neq f(1)$', 'error', and 'fail'. When the result is one of the first two outputs, that statement is guaranteed to be true, and the probability of obtaining 'error' can be made arbitrarily small and thus neglected. However, the probability of the algorithm's returning 'error' is $\frac{1}{2}$, in which case nothing is learned about the relationship between $f(0)$ and $f(1)$, and the algorithm must be run again. With repeated errors, it might actually turn out that the quantum computer evaluates f *more* than twice in order to answer the question in some instances, although since there is essentially a fifty per cent success rate, the expected number of attempts required is two.

1.2.1 Quantum algorithms

Deutsch's algorithm provides an example of a problem described under the quantum query model, in which information about the problem to be solved can be obtained through calls to a so-called oracle. In this case the oracle has complete knowledge of the function f, but only returns one of its outputs at a time. The fewer the number of required calls to the oracle, the more efficient the algorithm. Due to the probabilistic nature of measurement outcomes in quantum mechanics, it is not uncommon for a quantum algorithm to provide an answer to a question with only some probability less than unity, as does Deutsch's, but this is not a generic feature. The function f used in Deutsch's algorithm has a fixed input size of one bit. More generally, algorithms are expected to solve problems with a variable input size, specified by n classical bits. In such cases the algorithm defines a family of solutions—a family of circuits, in the case of the circuit model discussed in the following section—such that for any allowed value of n, there exists a member of the family that solves any instance of the problem with that input

length.

In 1992 Deutsch and Jozsa provided the first example of a quantum algorithm that both yields a correct answer with certainty and is more efficient than any classical solution to the same problem, in that it requires fewer queries to the designated oracle [10]. While interesting from the point of view of computational complexity, in that this result hints at the possibility that the power of quantum computers may exceed that of classical ones, the problem that they solve is of little practical interest. Furthermore it turns out that although the Deutsch–Jozsa algorithm is exponentially more efficient than any deterministic classical algorithm for the same task, it requires two oracle calls and there exists a non-deterministic classical algorithm that is expected to take approximately three queries. The quantum algorithm provides an absolute speed-up, but not a significant one.

Two years later, Shor presented a quantum algorithm capable of factoring an integer exponentially more quickly than the fastest known classical algorithm [11]. This is of great practical interest as a wide range of modern cryptographic protocols, including the RSA encryption used to secure many online transactions [12], are secure only so long as it remains difficult to factor large numbers. While this may sound like a coup for quantum computers (and in many ways it is), theoretically speaking the difficulty of the factoring problem is presently unknown—no classical algorithm yet exists that is as efficient as Shor's, but neither has it been shown impossible that one could be developed. Therefore Shor's algorithm provides no obvious theoretical distinction between the powers of classical and quantum computers, despite being one of the most exciting practical results to emerge. Finally, after a further two years, Grover provided a quantum algorithm that can search an unordered database in quadratically less time than the best possible classical algorithm is expected to take [13]. This result is potentially somewhat more useful practically, but any database which is going to be accessed many times can be classically indexed so as to be subsequently searched exponentially more quickly than by

Grover's algorithm. Such an indexing process is comparable to the task of sorting the list which, as discussed in Section 1.1.1, takes $O(N \log N)$ operations. Grover's search runs in a time $O(\sqrt{N})$, so creating an index and searching classically becomes the more efficient option if the database is to be searched more than $\Omega(\sqrt{N})$ times. Nevertheless, the specified problem is of some practical interest, and Grover's algorithm does provide an absolute, if only polynomial, speed-up over classical computers. The existence of these algorithms was sufficient motivation to fuel the rapid growth seen since the turn of the century in the fields of quantum computation and quantum information science.

1.3 Quantum bits and the quantum circuit model

As in the classical case, several models for quantum computation have been proposed and shown to be equivalent to Deutsch's universal quantum Turing machine. Perhaps the most common of these, and the one to which the quantum walk schemes of subsequent chapters are compared, is the *quantum circuit model*, so called in direct analogy to the circuit model of classical computation. A brief overview of another model of quantum computation can be found in Section 1.6, and the quantum walk model is discussed in detail throughout subsequent chapters. Two properties are possessed by each of these models, and indeed of every model of quantum computation. The first is that any quantum algorithm can be efficiently recast to run under the model, and the second is that conversely any instance of the model can be simulated efficiently by a universal quantum computer, which is often taken to be described by the circuit model.

Figure 1.2 provides a simple example of a quantum circuit, using some of the gates described in the current section. There are many similarities between the two circuit models, quantum and classical, including the use of wires to carry information and gates to transform it, and the ability to parallelize gates that act on disjoint subsets of the

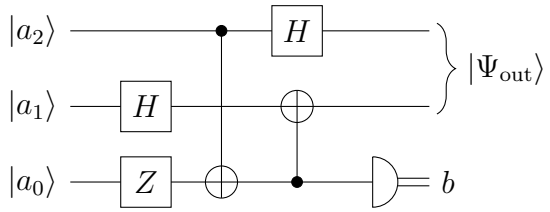


Figure 1.2: Example quantum circuit on three qubits. The symbols $-\boxed{H}-$ and $-\boxed{Z}-$ represent the Hadamard and Pauli-Z gates respectively, while the vertical connection between rails is a CNOT gate with the target qubit specified by $\bigoplus$ and the control by $\bullet$. The half-disc at the end of the lowest rail represents a measurement of that qubit, which results in the classical bit b, carried by a classical wire, while the two upper qubits exit the circuit in the two-qubit state $|\Psi_{\text{out}}\rangle$.

wires at the same step of the algorithm.

Just as the classical bit can take on one of two mutually exclusive values, the fundamental unit of quantum information is defined in terms of two orthogonal states. However as a quantum element, the quantum bit, or *qubit*, is able to take on any normalized state in the two-dimensional Hilbert space $\mathbb{C}^2$ spanned by the computational basis states $|0\rangle$ and $|1\rangle$ and thus exist in an arbitrary superposition of the two values. The standard representation of these states as *ket* vectors in $\mathbb{C}^2$ is

$$|0\rangle \doteq \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle \doteq \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.5}$$

Their associated dual vectors are the *bras*

$$\langle 0| \doteq \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad \langle 1| \doteq \begin{pmatrix} 0 & 1 \end{pmatrix}, \tag{1.6}$$

and an inner product is given by a *Dirac bracket*, such as $\langle i|j\rangle$. Similarly, the outer product $|i\rangle\langle j|$ yields a matrix.

Since $|\psi\rangle$ is required to be normalized, a useful representation of the state is to set $\alpha = \cos(\frac{\theta}{2})$ and $\beta = \mathrm{e}^{\mathrm{i}\phi}\sin(\frac{\theta}{2})$ for real numbers $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$, which leads

to the common description of $|\psi\rangle$ as the point on a unit sphere, referred to as the unit 'Bloch' sphere, with polar angle θ and azimuthal angle ϕ. The computational basis states $|0\rangle$ and $|1\rangle$ correspond to the north and south poles of the Bloch sphere. The coefficient α can be chosen to be real without loss of generality, because the overall phase of any quantum system has no effect on its measurement statistics, as shown in the next section.

1.3.1 Measuring a qubit

While a classical bit can take on either the state $b = 0$ or $b = 1$, the state of a qubit is a two-dimensional vector with complex coefficients, and thus can take on any of a continuum of states, expressed as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.7}$$

with $|\alpha|^2 + |\beta|^2 = 1$. Any quantum state such as this that can be expressed as a single vector in a Hilbert space is referred to as *pure*; one can also consider *mixed* states, which correspond to classical probability distributions over pure states, but doing so does not increase the computational power of the circuit model, so for convenience attention is restricted here to pure states only. In order for a qubit to reveal information—the result of a computation, for example—to the classical world, it must be *measured*.

The measurement postulate of quantum mechanics states that to every classically observable property of a quantum mechanical system there is associated a Hermitian operator, the eigenvalues of which are the possible outcomes of the measurement. After the measurement, the state of the system is given by the eigenvector corresponding to the measurement result. An immediate consequence is that even though the set of allowed states of a qubit is a continuum, as a two-dimensional system the classically accessible results of a single-qubit measurement form a dichotomy because the measurement op-

erator has only two eigenvectors. Given a Hermitian operator Λ with eigenvectors $|\lambda_i\rangle$ and corresponding eigenvalues λ_i, the probability of obtaining the outcome λ_i from the measurement of a system in the state $|\phi\rangle$ is $|\langle\lambda_i|\phi\rangle|^2$. It is for this reason that an overall phase on a quantum state is irrelevant. If θ is a real number and $|\phi\rangle$ is any quantum state, then $|\langle\lambda_i|\phi\rangle|^2 = |\langle\lambda_i|e^{i\theta}|\phi\rangle|^2$, so the states $|\phi\rangle$ and $e^{i\theta}|\phi\rangle$ are identical as far as measurements are concerned.

For a specific example of a measurement, consider a single qubit. The allowed outcomes of a measurement in the computational basis are 0 and 1, corresponding to the basis states $|0\rangle$ and $|1\rangle$. Constructing the operator

$$\Lambda = 0|0\rangle\langle 0| + 1|1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.8}$$

one finds that the single-qubit state $|\psi\rangle$ of Equation (1.7) has probability of $|\langle 0|\psi\rangle|^2 = |\alpha|^2$ of producing the outcome '0' and leaving the qubit in the state $|0\rangle$, and probability $|\langle 1|\psi\rangle|^2 = |\beta|^2$ of resulting in the output '1', with the qubit left in the state $|1\rangle$. A measurement in the computational basis is represented graphically in a circuit diagram by a half disc, with a quantum wire input on the left and a classical wire carrying a single bit value on the right, as depicted on the lowest rail of Figure 1.2.

Such a measurement is called *projective* because a state vector initially in some superposition of the eigenvectors of the measurement operator is projected onto a single one of them. A *projector* P is an operator such that $P^2 = P$, and is said to be orthogonal if additionally P is Hermitian, $P^\dagger = P$. Any set $\{P_i\}$ of orthogonal projectors that sum to the identity, $\sum_i P_i = I$, yields a projective measurement. If the system to be measured is in the state $|\psi\rangle$, then the measurement outputs result i with probability $p(i) = \langle\psi|P_i|\psi\rangle$,

after which the system is left in the state

$$\frac{1}{\sqrt{p(i)}} P_i |\psi\rangle. \tag{1.9}$$

Projective measurements are not the only measurement formalism within quantum mechanics, nor the most general, yet they are sufficient for universal computation under both the circuit model and the quantum walk models as discussed in the present work. They are thus the only form of measurement used herein.

1.3.2 Single-qubit computational gates

Measuring a qubit is only interesting if its state represents an interesting quantity, such as the result of a computation. In addition to measurements then, quantum-mechanical equivalents to classical logic gates are required. Under the circuit model, the set of such computational gates is exactly the set of unitary operations on qubits. Again, more general transformations of quantum states exist, yet under the circuit model universal computation is possible with only unitary gates; throughout this work the term 'gate' is synonymous with 'unitary operation'.

Classically there are exactly two logical operations that can map a single bit to a single bit: the identity, which accepts a bit value as input and returns the same value as output, and the bit flip, which accepts either 0 or 1 as input, then outputs the other. The discussion of operations that can be performed on a single qubit is rather longer however, as in general both input and output states can be of the form in Equation (1.7), though again it is important to note that the coefficients α and β cannot be recovered classically as output. Such a state can be used as input to a subsequent quantum gate, but at the end of the computation only a single bit of information, either a 0 or a 1, can be returned by a qubit.

A one-bit logic gate on a classical bit is given by a transformation $f : \{0,1\} \to \{0,1\}$ from the one-bit state space to itself. Similarly, a quantum logic gate on a single qubit is any transformation $F : \mathbb{C}^2 \to \mathbb{C}^2$ that preserves the vector norm. That is, given an input qubit state $|\psi_{\text{in}}\rangle$ and an output qubit state $|\psi_{\text{out}}\rangle = F|\psi_{\text{in}}\rangle$, whenever the input state satisfies $|\langle\psi_{\text{in}}|\psi_{\text{in}}\rangle| = 1$ it must also true that

$$|\langle\psi_{\text{out}}|\psi_{\text{out}}\rangle| = |\langle\psi_{\text{in}}|F^\dagger F|\psi_{\text{in}}\rangle| \stackrel{!}{=} 1. \tag{1.10}$$

This immediately specifies that the transformation F must be unitary, and so a general quantum logic gate is often labelled 'U'. The requirement that qubit gates be unitary leads to two important distinctions from the classical case, before any specific gates are investigated. Firstly, the no-cloning theorem implies that the classical FANOUT gate cannot be used to duplicate arbitrary quantum bits [14]. Nevertheless, it has been shown that a quantum version of FANOUT that copies only the computational basis states—i.e. classical bits—exactly is still a useful, and indeed powerful, quantum gate [15]. Secondly, many of the most common classical logic gates are many-to-one maps, and therefore non-invertible. The evolution of information through such gates is irreversible, and so not unitary. In spite of these functional differences from the classical gates described thus far, it is possible to implement universal classical computation in a reversible manner, a fact shown by Bennett in 1973 [16] and used by Deutsch in his development of the quantum Turing machine.

Any 2×2 unitary matrix in U(2) represents a valid quantum logic gate in the circuit model, however a few appear frequently and are deserving of particular note. Four of these are the *Pauli spin matrices* and the identity,

$$X \doteq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \doteq \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \quad Z \doteq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad I \doteq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{1.11}$$

These operators are Hermitian as well as unitary, and together they form a basis for the vector space of 2×2 Hermitian matrices; since they are Hermitian they generate additional unitary matrices under exponentiation.[3] This gives rise to the *rotation operators* given by

$$R_X(\theta) \doteq \mathrm{e}^{-\mathrm{i}X\theta/2} = \cos\left(\tfrac{\theta}{2}\right) I - \mathrm{i}\sin\left(\tfrac{\theta}{2}\right) X, \tag{1.12}$$

and likewise for R_Y and R_Z. Any two of these rotations can be used to provide a decomposition of an arbitrary single-qubit operation U with appropriate choices of rotation angles. If $U \in \mathrm{U}(2)$, then there exist real numbers ϕ, θ_1, θ_2, and θ_3 such that U can be expressed exactly as

$$U = \mathrm{e}^{\mathrm{i}\phi} R_X(\theta_1) R_Z(\theta_2) R_X(\theta_3). \tag{1.13}$$

Since a generic single-qubit unitary U maps $|\psi\rangle$ to another state vector $|\psi'\rangle = U|\psi\rangle$ and both $|\psi\rangle$ and $|\psi'\rangle$ are points on the Bloch sphere, U is also referred to as a rotation on the Bloch sphere, and can be expressed as $R_l(\gamma)$ for some axis l and angle γ.

Another single-qubit operator of note is the *Hadamard matrix*

$$H \doteq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{X+Z}{\sqrt{2}}, \tag{1.14}$$

which maps each computational basis state to an even superposition of the two.

1.3.3 Two-qubit gates

One of the postulates of quantum mechanics states that the Hilbert space of a composite system is tensor product of the state spaces of its constituents. In the case of qubits this means that a two-qubit state is described by a normalized vector in $\mathbb{C}^2 \otimes \mathbb{C}^2$, or more generally an n-qubit state by a normalized vector in $(\mathbb{C}^2)^{\otimes n}$. A logical operator on n

[3] In fact, the matrices $\{\mathrm{i}X, \mathrm{i}Y, \mathrm{i}Z\}$ span the Lie algebra $\mathfrak{su}(2)$ and thus generate the Lie group SU(2) of two-dimensional unitary matrices U with $\det(U) = 1$.

qubits is a unitary $U \in \mathrm{U}(2^n)$; again, this ensures that normalized states are mapped to normalized states. It turns out, in a manner formalized in the following section, that any unitary operation on n qubits can be decomposed exactly into a sequence of one- and two-qubit operations. As with single-qubit gates, there are certain two-qubit gates that appear frequently and as such are worth introducing explicitly here. The controlled-NOT (or controlled-X) and controlled-phase gates are given by

$$\textsc{cnot} \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \textsc{cphase}(\phi) \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{\mathrm{i}\phi} \end{pmatrix} \tag{1.15}$$

respectively, in the canonical two-qubit computational basis

$$|00\rangle \doteq \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle \doteq \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle \doteq \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle \doteq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{1.16}$$

Here the standard abbreviated form of the tensor product notation has been introduced, wherein

$$|ij\rangle \doteq |i\rangle \otimes |j\rangle. \tag{1.17}$$

These gates are referred to as 'controlled' because they either do or do not enact a single-qubit gate (here NOT or PHASE) on the second 'target' qubit, depending on whether the state of the first 'control' qubit is in the state $|1\rangle$ or $|0\rangle$.

The importance of such gates, indeed of almost all two-qubit operations, is their ability to generate *entanglement*, which is a uniquely quantum-mechanical feature of composite systems. Classically, if a system comprises two subsystems then the allowed states of the system as a whole are simply the set of all possible pairs of allowed states of the subsystems. Quantum mechanically, it may also be the case that each subsystem

is in a well-defined single-qubit state, as for example in the two-qubit state

$$|\Psi\rangle = (\alpha|0\rangle_1 + \beta|1\rangle_1) \otimes (\gamma|0\rangle_2 + \delta|1\rangle_2) = |\psi_1\rangle \otimes |\psi_2\rangle. \tag{1.18}$$

Any two-qubit state that can be factored into a tensor product of two one-qubit states like this, with appropriate choices for α, β, γ, and δ, is called *separable* or referred to as a *product state*. However, there exist states in the Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2$ that are not of this form. Consider

$$|\Phi^+\rangle \doteq \frac{1}{\sqrt{2}} \left(|0\rangle_1 \otimes |0\rangle_2 + |1\rangle_1 \otimes |1\rangle_2\right) = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right), \tag{1.19}$$

one of the four maximally entangled Bell states on two qubits. There is no way to factorize this state into the form of Equation (1.18), and furthermore it can be shown that there exists no basis in which the state factorizes, meaning that it is not possible to describe the states of the constituent systems individually. They exist only in a collective *entangled* state.

The physical interpretation of this becomes apparent when one considers performing a measurement in the computational basis on, say, the first qubit subsystem. If the outcome of the measurement is $b \in \{0, 1\}$, then immediately afterwards the state of the system as a whole is $|b\rangle \otimes |b\rangle$, in which the second qubit is in the well-defined state $|b\rangle$. That is, a measurement on one component of the system yields knowledge about the state of another. The crucial aspect to the phenomenon of entanglement is not simply this correlation however; if the initial state before measurement were instead $|\Psi_{\text{prod}}\rangle \doteq |00\rangle$, then immediately after measuring the first qubit in the computational basis and obtaining outcome 0, the second qubit would also be known to be in the state $|0\rangle$, but this would not be interesting. Suppose that instead a measurement were made in the $|\pm\rangle$ basis

defined by

$$|\pm\rangle \doteq \frac{1}{\sqrt{2}}\left(|0\rangle \pm |1\rangle\right). \tag{1.20}$$

In this basis the state $|\Psi_{\text{prod}}\rangle$ is given by

$$|\Psi_{\text{prod}}\rangle = \frac{1}{\sqrt{2}}\left(|+\rangle_1 + |-\rangle_1\right) \otimes \frac{1}{\sqrt{2}}\left(|+\rangle_2 + |-\rangle_2\right) = \frac{1}{2}\left(|{+}{+}\rangle + |{+}{-}\rangle + |{-}{+}\rangle + |{-}{-}\rangle\right) \tag{1.21}$$

and thus a measurement of the first outcome that yields, for example, '+' leaves the system in the state

$$\frac{1}{\sqrt{2}}\left(|{+}{+}\rangle + |{+}{-}\rangle\right) = |+\rangle_1 \otimes \frac{1}{\sqrt{2}}\left(|+\rangle_2 + |-\rangle_2\right). \tag{1.22}$$

So learning the state of the first qubit in this basis reveals nothing about the state of the second, which has an equal chance to be found in either of the $|\pm\rangle$ states if measured itself.

On the other hand the entangled state $|\Phi^+\rangle$ appears in this transformed basis as

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\left(|{+}{+}\rangle + |{-}{-}\rangle\right). \tag{1.23}$$

A measurement of the first qubit in this basis that yields an outcome of '+' leaves the system in the state $|{+}{+}\rangle$, and similarly an outcome of '−' results in $|{-}{-}\rangle$. The states of the two qubits are correlated in this basis as well, and in fact in every basis.

The importance to quantum computation of two-qubit gates such as CNOT and CPHASE is their ability to act on a product state and transform it into an entangled one, at least for certain input product states. For example, the CPHASE gate acts trivially on each of the two-qubit computational basis states in Equation (1.16), doing nothing to three of them and merely adding an unimportant overall phase to $|11\rangle$. However, the

input state

$$|\Psi_{\text{in}}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_1 + |1\rangle_1\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle_2 + |1\rangle_2\right) = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right) \tag{1.24}$$

is transformed into

$$|\Psi_{\text{out}}\rangle = \textsc{cphase}(\phi)|\Psi_{\text{in}}\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + \mathrm{e}^{\mathrm{i}\phi}|11\rangle\right), \tag{1.25}$$

which is an entangled state for every value of ϕ that is not an integral multiple of 2π.

Another two-qubit gate that is important to many schemes for quantum computation, despite not being entangling, is the SWAP gate, which exchanges to qubits. In any quantum computer qubits will be encoded in physical elements of a system, and it may be that however these physical qubits are arranged, they can each only interact with adjacent qubits. This would mean that, for example, the first CNOT gate in Figure 1.2 could not be performed because qubits 1 and 3 are not adjacent. To circumvent this problem the nearest-neighbour SWAP gate, defined in the two-qubit computational basis as

$$\textsc{swap} \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1.26}$$

can be used to interchange qubits 2 and 3 both before and after enacting CNOT on the qubits in positions 1 and 2.

Denote by $\textsc{cnot}_{ij}$ the gate that implements a single-qubit NOT operation on the target qubit j contingent upon the state of the control qubit i, and let $\textsc{swap}_{ij}$ be the operation that exchanges qubits i and j. If only nearest-neighbour operations are permitted—that is, if these gates are restricted to cases in which $j = i \pm 1$—then the $\textsc{cnot}_{13}$ gate in

Figure 1.2 can be replaced by the sequence of operations

$$\text{CNOT}_{13} = \text{SWAP}_{23}\text{CNOT}_{12}\text{SWAP}_{23}. \tag{1.27}$$

Thus SWAP is a useful gate, and one that features prominently in Chapter 6, despite its not being required for the theoretical description of universal quantum computation described next.

1.4 Universal quantum computation under the circuit model

The state space of even a single qubit is a continuum, and thus uncountably infinite. Nevertheless, it has been shown that remarkably there exist finite sets of unitary gates capable of efficiently approximating any given gate to arbitrary accuracy.

A quantum computation in its simplest terms is a unitary operation U_f that encodes a function which transforms a given n-qubit input state into a state that is to be measured in order to determine the outcome of the computation. In looking for a quantum equivalent of the concept of a universal gate, the question is, given such a U_f, how can it be decomposed into a sequence of operations from a finite set? The first step toward answering this question is to consider only those U_f that act on a single qubit, after introducing the concept of an ε-approximation to an arbitrary unitary operation.

Definition 1.1 (Cf. Reference [17])**.** *A unitary operation V, possibly comprising a sequence S of gates, provides an* ε-approximation *to a quantum gate U if*

$$||U - V|| \doteq \sup_{|||\psi\rangle||=1} ||(U - V)|\psi\rangle|| < \varepsilon. \tag{1.28}$$

That is, taken over all normalized input states $|\psi\rangle$, the magnitude of the largest difference between $U|\psi\rangle$ and $V|\psi\rangle$ is less than ε. The concept of approximating a desired

gate to arbitrary accuracy without necessarily executing it exactly is what allows a finite set of gates to be considered universal for a continuum of operations, as made concrete by the following definition.

Definition 1.2 (Cf. Definition 4.3.4 of Reference [18])**.** *A set of gates is said to be* universal for single-qubit gates *if a sequence of gates from the set can provide an ε-approximation to any single-qubit unitary gate.*

This definition is primarily useful in conjunction with the following theorem, which provides a sufficient criterion for determining whether a set of gates is universal for single-qubit gates.

Theorem 1.1 (Theorem 4.3.5 of Reference [18])**.** *If a set of two single-qubit gates (rotations) $\mathcal{G} = \{R_l(\beta), R_m(\gamma)\}$ satisfies the conditions*

1. *l and m are non-parallel axes of the Bloch sphere, and*
2. *$\beta, \gamma \in [0, 2\pi)$ are real numbers such that $\frac{\beta}{\pi}$ and $\frac{\gamma}{\pi}$ are not rational*

then $\mathcal{G}$ is universal for single-qubit gates.

Theorem 1.1 shows that there indeed exist finite gate sets that are universal for single-qubit computation, but it does not speak to the efficiency with which an arbitrary operation can be simulated by the universal set, in terms of the number of times the elements of the set must be applied in order to approximate the desired outcome. The amazing fact that it is indeed possible for a finite set of gates to efficiently approximate arbitrary single-qubit operations is the result of the Solovay-Kitaev theorem, stated below as Theorem 1.2 in terms of the following definition.

Definition 1.3 (Cf. Definition 1 of Reference [17])**.** *An* instruction set *$\mathcal{G}$ for single-qubit operations is a set of quantum gates satisfying:*

1. *The number of gates in the set, $|\mathcal{G}|$, is finite.*
2. *All gates $g \in \mathcal{G}$ are in* SU(2)*; that is, they are unitary with determinant 1.*
3. *For each $g \in \mathcal{G}$, its inverse operation $g^\dagger$ is also in $\mathcal{G}$.*
4. *$\mathcal{G}$ is universal for single-qubit gates.*

Theorem 1.2 (Solovay-Kitaev; cf. Theorem 1 in Reference [17])**.** *Let $\mathcal{G}$ be an instruction set for* SU(2)*, and a desired accuracy $\varepsilon > 0$ be given. Then there is a constant $c \approx 2$ such that for any $U \in$ SU(2) there exists a finite sequence S of gates from $\mathcal{G}$, of length $O(\log^c(1/\varepsilon))$, such that S provides an ε-approximation to U.*

It is the logarithmic dependency of the length of the approximating sequence on the desired error tolerance that justifies the statement that the approximation is efficient.

1.4.1 Computing with two qubits

Local operations cannot generate entanglement, so it is clear that at least one two-qubit gate must be part of any universal set of quantum gates. The following result, due to Zhang *et al.* [19], shows that a single two-qubit gate (possibly executed multiple times) in conjunction with a universal set of single-qubit gates can efficiently generate any two-qubit gate.

Theorem 1.3. *Let U_C be either* CNOT*, or* CPHASE(ϕ) *with $\phi \in [\frac{\pi}{2}, \pi]$. Given U_C and access to arbitrary single-qubit operations, any two-qubit gate $U \in$ SU(4) can be simulated exactly with at most six applications of U_C, and seven local gates.*

Here a local gate is a two-qubit gate of the form $U_A \otimes U_B$, where $\{U_A, U_B\} \in$ SU(2) are single-qubit gates. If access to arbitrary exact single-qubit gates is not available, each of the local gates must be approximated from an instruction set, in a sequence as

in the Solovay-Kitaev theorem. This raises the question of whether a sequence of well-approximated gates is itself well approximated, which is answered in the affirmative by the chaining inequality of the following lemma.

Lemma 1.1 (Cf. Equation (4.63) of Reference [4])**.** *If a sequence of unitary operators $V_1, \ldots, V_m$ is used to approximate a desired sequence $U_1, \ldots, U_m$, then the individual errors add at most linearly:*

$$||(U_m \cdots U_1) - (V_m \cdots V_1)|| \leq \sum_{i=1}^{m} ||U_i - V_i||. \tag{1.29}$$

This means that to simulate a sequence of 14 single-qubit gates—the most required by Theorem 1.3—to an accuracy of ε, each gate should individually be simulated to an accuracy of $\varepsilon/14$, leading immediately to the following corollary.

Corollary 1.1. *Let U_C be either* CNOT*, or* CPHASE(ϕ) *with $\phi \in [\frac{\pi}{2}, \pi]$. If U_C, a single-qubit instruction set $\mathcal{G}$, and an accuracy $\varepsilon > 0$ are given, then there is a constant $c \approx 2$ such that for any two-qubit gate $U \in \mathrm{SU}(4)$ there exists a finite sequence S of gates from $\mathcal{G} \cup \{U_C\}$ of length $O(\log^c(14/\varepsilon))$ such that S provides an ε-approximation to U.*

Therefore arbitrary two-qubit gates can be simulated efficiently under the same conditions for which single-qubit operations can, when a single appropriately chosen entangling gate is added to the instruction set.

1.4.2 Many qubits

The next step is to consider a computation on n qubits. Theorem 1.1 states that a finite set of one-qubit gates can be universal for single-qubit computation; Corollary 1.1 likewise shows that a finite set containing one- and two-qubit gates can be universal for two-qubit operations, if at least one two-qubit gate in the set cannot be written as a tensor product of one-qubit gates. One might wonder then whether a computation on n

qubits in general requires at least one operation that cannot be decomposed into a tensor product of lower-dimensional gates. It turns out that this is not the case, and indeed the gate sets already discussed are universal for quantum computation on an arbitrary number of qubits.

What does not carry over from two qubits to an n system is the ability to *efficiently* simulate arbitrary operations—there exist unitary operations on n qubits that require at least $\Omega(2^n \log(1/\varepsilon)/\log(n))$ gates for an ε-approximation by any finite gate set [4]. Any finite gate set will have a 'largest' gate that acts on n_0 qubits at once, yet there exist unitary operations on n qubits for all $n > n_0$. As n increases, the dimension of the space in which the gates must act grows exponentially, while the gate size remains fixed. This result should not be discouraging though, for the same is true classically and nevertheless classical computers have proven themselves immeasurably useful. The important point is not that some computations are difficult to perform, but rather that certain particularly advantageous computations are not. Of those discovered to date perhaps the most promising is the quantum Fourier transform, which plays a central role in Shor's factoring algorithm and which can be efficiently decomposed into one- and two-qubit gates for any input size n. For the purposes of the present work however, it is sufficient to know that such applications exist, and that any set of gates satisfying the requirements of Corollary 1.1 is universal for quantum computation.

1.5 Quantum error correction

Classical computers generally only require error correction when communicating over large distances that lead to significant signal attenuation. Bits are stored in physical systems that are enormous on the quantum scale. The states of those systems that encode the bit values 0 and 1 are so disparate that the probability of a bit's 'flipping'

from its assigned value to the other, erroneous, value due to external noise such as mechanical vibrations or thermal fluctuations is negligible. The state of a qubit on the other hand can be easily and significantly altered by any external coupling, including to other nearby qubits when no multi-qubit gate is being enacted. If such noise alters the state of a qubit during a computation, an error is introduced because the encoded value is no longer correct.

A straightforward tactic to counter bit-flip errors during classical communication is to implement a repetition code, in which the bit values 0 and 1 are encoded in longer bit strings as $00\cdots0$ and $11\cdots1$. The strings encoding the computational bits are called code words, and if at a later time it is found that one of the bit strings is no longer equal to either code word, then resetting all bits to the majority value of the string is likely to fix the error. For example the three-bit repetition code is given by $\underline{0} = 000$ and $\underline{1} = 111$, where an underlined value represents an encoded computational, or logical, bit, while the non-underlined values correspond to values of physical bits used in the encoding. Given a relatively trustworthy communications architecture in which it is reasonable to assume that bits are more likely to remain correct than to flip, if it is found that a supposedly encoding bit string is equal to 001, it is reasonable to conclude that the encoded bit should be $\underline{0}$. Flipping the third encoding bit back to 0 corrects the error.

Unfortunately, three factors prevent such a code from being directly ported over to the quantum domain to protect encoded qubits. One is that the no-cloning theorem prevents the copying of arbitrary qubit states to create a repetition in the first place. Another is that even were such an encoding possible, the act of measuring a qubit to determine whether its state has changed, itself alters the state and the information it encodes is lost anyway. Finally, qubit states can be deformed continuously and thus there is a continuum of errors that can occur. Amazingly, there nevertheless exist techniques for overcoming these problems; this is the realm of quantum error correction and quantum

error-correcting codes. A thorough exposition of the many significant results that have shown how to solve these issues is beyond the scope of this thesis. However, that they can be solved is of fundamental importance to any extensive discussion of quantum computing, and the subject appears again in Chapter 6, wherein the first (to my knowledge) construction of a quantum-walk scheme for error correction is presented. A brief introduction is thus included here, and for more in-depth presentations the interested reader is referred to a review article by Gottesman [20], and Chapters 5 and 10 respectively of the textbooks by Mermin [21] and Nielsen and Chuang [4], from which the following results are taken.

The no-cloning theorem states that it is impossible to duplicate an arbitrary quantum state, but does not state that cloning a known state is forbidden. That is, while an unknown state $|\psi\rangle$ and fiducial state such as $|0\rangle$ cannot reliably be transformed from $|\psi\rangle|0\rangle$ into $|\psi\rangle|\psi\rangle$, the creation of states such as $|0\rangle|0\rangle|0\rangle$ and $|1\rangle|1\rangle|1\rangle$ (or even $|\psi\rangle|\psi\rangle$, if $|\psi\rangle$ is specified) is allowed. This means that there is at least hope for the creation of code words corresponding to computational basis states. With respect to the continuum of possible unitary errors that a qubit can undergo, amazingly it suffices to be able to detect and correct only the undesired action of the Pauli matrices. This is a consequence of the fact that an arbitrary unitary operation can be expressed as a sequence of Pauli rotations as in Equation (1.13), and that in turn these rotations can be written as a linear combination of the identity and a Pauli matrix as in Equation (1.12).

The first of the errors that must be corrected is an undesired X operation, which turns $|0\rangle$ into $|1\rangle$ and vice versa, and is thus analogous to the classical bit flip error. A uniquely quantum-mechanical error is the phase flip, which negates the relative phase between the $|0\rangle$ and $|1\rangle$ components of a qubit; this corresponds to the action of Z. The remaining Pauli matrix is $Y = \mathrm{i}XZ$, which is therefore a combination of the bit- and phase-flip errors.

1.5.1 Three-qubit codes

It turns out that the straightforward encoding of the computational basis states as in the classical repetition code, i.e. as

$$|\underline{0}\rangle = |000\rangle, \quad |\underline{1}\rangle = |111\rangle, \tag{1.30}$$

can be used to detect and correct a single X error. Under this code, an encoded qubit in the state $|\psi\rangle = \alpha|\underline{0}\rangle + \beta|\underline{1}\rangle$ corresponds to an encoding state of $\alpha|000\rangle + \beta|111\rangle$, which is distinct from a tensor product three qubits each in the state $\alpha|0\rangle + \beta|1\rangle$ (unless either α or β vanishes). The encoding of a single-qubit state $\alpha|0\rangle + \beta|1\rangle$ is accomplished with two ancillary qubits in the fiducial $|0\rangle$ state and two CNOT operations. With CNOT_{ij} a CNOT gate affecting the target qubit j based on the state of control qubit i, the encoding proceeds as

$$\begin{aligned} \text{CNOT}_{13}\text{CNOT}_{12}(\alpha|0\rangle + \beta|1\rangle)|00\rangle &= \text{CNOT}_{13}\text{CNOT}_{12}(\alpha|000\rangle + \beta|100\rangle) \\ &= \text{CNOT}_{13}(\alpha|000\rangle + \beta|110\rangle) \\ &= \alpha|000\rangle + \beta|111\rangle. \end{aligned} \tag{1.31}$$

Suppose three physical qubits have been used to encode a logical qubit of the form $|\psi\rangle$, and after some time at most one of the qubits has undergone a bit flip. A *syndrome measurement* is a measurement that can determine when and where a particular error has occurred, without collapsing the encoded state and thus losing the quantum information. A bit flip can be diagnosed under the three-qubit bit-flip code by measuring the four

projectors

$$P_0 \doteq |000\rangle\langle 000| + |111\rangle\langle 111|, \tag{1.32a}$$

$$P_1 \doteq |100\rangle\langle 100| + |011\rangle\langle 011|, \tag{1.32b}$$

$$P_2 \doteq |010\rangle\langle 010| + |101\rangle\langle 101|, \tag{1.32c}$$

$$P_3 \doteq |001\rangle\langle 001| + |110\rangle\langle 110|, \tag{1.32d}$$

where P_i detects a bit flip on qubit i and a measurement result corresponding to P_0 indicates that no (bit-flip) error has occurred. For example, suppose that the third qubit has been flipped. The physical state is therefore $|\tilde{\psi}\rangle = \alpha|001\rangle + \beta|110\rangle$, and a measurement of the P_i returns 3 with certainty, since $\langle\tilde{\psi}|P_3|\tilde{\psi}\rangle = 1$. After the projective measurement, the three qubits remain in the state $|\tilde{\psi}\rangle$, so applying the gate $I \otimes I \otimes X$ transforms $|\tilde{\psi}\rangle \mapsto |\psi\rangle$, and the error has been corrected. Similarly, measurement results of 1 and 2 indicate errors that can be corrected by applying $X \otimes I \otimes I$ and $I \otimes X \otimes I$, respectively, while an outcome of 0 shows that no qubit has been bit flipped.

A phase-flip error can be corrected in a similar manner when one observes that in the $|\pm\rangle \doteq \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$ basis a phase flipping Z operator maps $|\pm\rangle \mapsto |\mp\rangle$, which is effectively a bit flip in this rotated basis. The code words $|\underline{0}\rangle = |{+}{+}{+}\rangle = H\otimes H\otimes H|000\rangle$, and $|\underline{1}\rangle = |{-}{-}{-}\rangle = H \otimes H \otimes H|111\rangle$ and equivalently defined syndrome measurement operators of the form in Equation (1.32), will identify and correct a single phase-flip error.

In practice such projections onto highly entangled states are difficult to implement. An alternative formulation of the three-qubit code achieves the same error-correcting results, while requiring only single-qubit measurements. To accomplish this without destroying the coherence between the three qubits of the code word, two further ancillary qubits are introduced. Figure 1.3 shows a quantum circuit that encodes a single-qubit

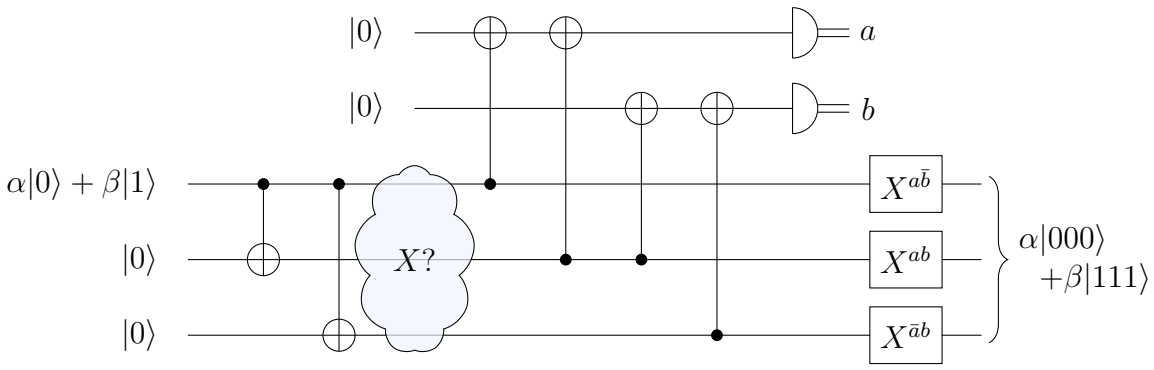


Figure 1.3: Quantum circuit to detect and correct at most one bit-flip error using the three-qubit code. The first two CNOT gates transform the input state $\alpha|0\rangle + \beta|1\rangle$ and two qubits in the fiducial $|0\rangle$ state into the three-qubit state $\alpha|000\rangle + \beta|111\rangle$. At most one X, or bit-flip, error might then occur. The ancillary qubits on the upper rails are entangled with the possibly corrupted three-qubit state, and measured individually. The outcomes a and b of these measurements specify which qubit must be corrected with an additional X gate; here $\bar{a} \doteq \text{NOT}\, a$. The final output is the original encoded state.

state $|\psi\rangle$, and then after at most one X error has occurred entangles two ancillary qubits with the encoded state to detect and correct the error.

The encoded state $\alpha|\underline{0}\rangle + \beta|\underline{1}\rangle$ is constructed by the first two CNOT gates from a single qubit in the input state $\alpha|0\rangle + \beta|1\rangle$ and two qubits in the fiducial $|0\rangle$ state, according to Equation (1.31). At some point after the encoding, at most one bit flip occurs in the state. That is, the three-qubit state is subject to one of the unitary operators $\{XII, IXI, IIX, III\}$, but since this is an error it is unknown which of these is applied. To identify the unknown operation, the ancillary qubits on the upper two rails are entangled with the possibly corrupted qubits. If no error occurs, i.e. the intervening operation is III, then the $|000\rangle$ term of the encoded state leaves the ancillary qubits in their initial state, $|00\rangle$, and the $|111\rangle$ term flips each of them twice—which is equivalent to not flipping them at all. Thus when no error occurs, the measurement outcomes a and b will both be 0, and since $X^0 = I$, the final three gates leave the intact state untouched.

Suppose instead that the operation XII is applied, flipping the first qubit of the

encoding state and leaving the erroneous state $\alpha|100\rangle + \beta|011\rangle$. Each of these terms causes the upper ancillary qubit to be flipped once, and the lower to be flipped twice. The classical outputs are thus $(a, b) = (1, 0)$, leading to $a\bar{b} = 1$ so that the first encoding qubit is flipped, correcting the error. A similar analysis shows that the errors IXI and IIX lead to the outputs $(a, b) = (1, 1)$ and $(a, b) = (0, 1)$, respectively, and the error in a physical qubit is corrected in each case, with no need to decode the logical qubit. It is important to note that while this procedure reveals which bit, if any, is flipped, it yields no information about the parameters α and β, and thus the encoded quantum information remains intact. This procedure of introducing $n - 1$ ancillary qubits and entangling each of them with a subset of the qubits in an n-qubit code applies to quantum error-correcting codes in general. It allows syndrome measurements to be performed as single-qubit measurements, and without decoding the logical qubits.

1.5.2 The nine-qubit Shor code

Correcting for both bit- and phase-flip errors requires longer code words, but can nevertheless still be accomplished. Shor proposed a nine-qubit code in which the code words of the phase-flip code are themselves encoded under the bit-flit code, and showed that this scheme is capable of correcting *arbitrary* single-qubit errors [22]. The first stage of the encoding is the same as the three-qubit phase-flip code, with $|0\rangle \mapsto |{+}{+}{+}\rangle$ and $|1\rangle \mapsto |{-}{-}{-}\rangle$. The second stage is to encode each of these three qubits using the three-qubit bit-flip code, leading to the encoding

$$|\underline{0}\rangle = \frac{1}{2\sqrt{2}}\left[(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)\right], \tag{1.33a}$$

$$|\underline{1}\rangle = \frac{1}{2\sqrt{2}}\left[(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)\right]. \tag{1.33b}$$

This process of nesting codes, wrapping the already encoded logical qubits of one code into a second layer of encoding under another, is called *concatenation*. Concatenation is particularly important not only because it can increase the number of single errors that a code can protect against, but also because the threshold theorem of quantum error correction guarantees that an error-prone quantum computer can efficiently simulate an ideal quantum computer so long as the errors are random, and below a certain threshold [23]. A certain class of codes referred to as 'fault tolerant' are able to be concatenated repeatedly, decreasing the error level at each stage until it is below the threshold.

1.5.3 The seven-qubit Steane code

That the continuum of single-qubit errors can be protected against by detecting and correcting a finite set of errors—namely X, Z, and XZ—is impressive. There are, however, inherent difficulties associated with the implementation of the three- and nine-qubit codes. Specifically, enacting single-qubit gates on the logical qubits can require the use of highly non-local physical gates on the underlying physical qubits. The seven-qubit Steane code mitigates this issue while still correcting arbitrary single-qubit errors. Furthermore the Steane code is *fault tolerant*, meaning that operations can be performed on encoded logical qubits without decoding them, and a single error in the procedure cannot propagate to more than one qubit. The three- and nine-qubit codes are not fault tolerant.

Introduced by Steane [24], the seven-qubit Steane code is a so-called CSS code, named for the developers Calderbank and Shor [25], and independently Steane [26], of this class of codes. Its code words are

$$|\underline{0}\rangle = \frac{1}{2\sqrt{2}}\big(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle\big) \tag{1.34a}$$

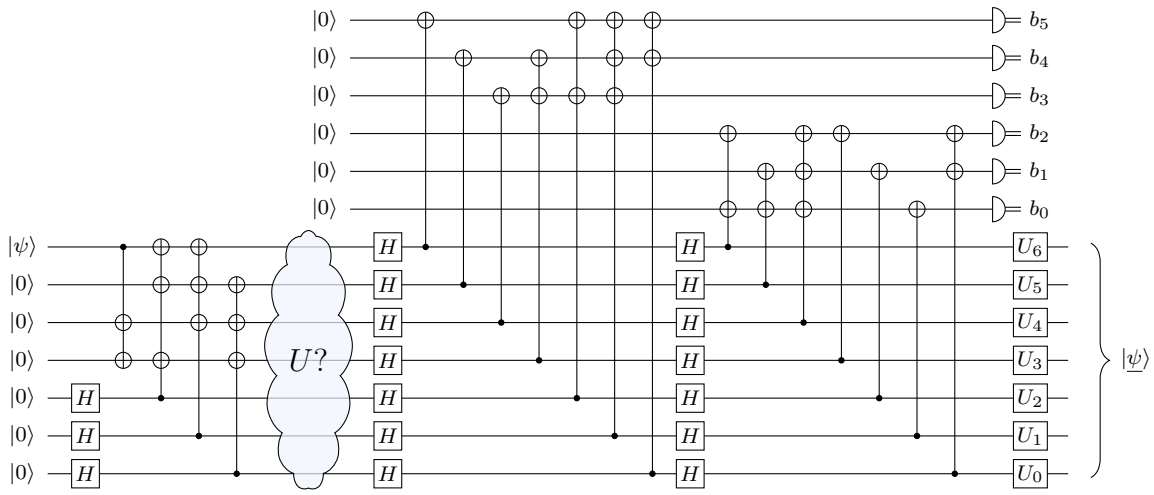


Figure 1.4: Quantum circuit for the seven-qubit Steane code. The input state $|\psi\rangle$ on the middle rail is encoded into the logical state $|\underline{\psi}\rangle$ by the first set of Hadamard and CNOT operations, after which an unknown unitary error may or may not occur on a single qubit. The next two sets of Hadamards and CNOTs entangle the ancillary qubits of the upper six rails with the now-unknown seven-qubit state, after which a set of six single-qubit measurements yield a set of bit values specifying which corrections U_i are to be made, if any.

and

$$|\underline{1}\rangle = \frac{1}{2\sqrt{2}}\big(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle\big), \tag{1.34b}$$

The circuit to encode a logical qubit, perform a syndrome measurement, and correct a diagnosed error is identical in spirit to the three-qubit version in Figure 1.3, but contains four additional physical encoding qubits and four more ancillary qubits. It can be seen in Figure 1.4.

Gates such as the Hadamard and Pauli operators can be implemented on the encoded qubits with local operations on the encoding qubits. For example, a Hadamard operation on a logical qubit is simply $\underline{H} = H^{\otimes 7}$ and a logical CNOT gate can similarly be

implemented as $\underline{\text{CNOT}} = \text{CNOT}^{\otimes 7}$. Gates for which this is true are called *transversal*, and can be implemented fault tolerantly. The transversal property is not inherent to a gate, but rather applies to a gate with respect to a given error-correcting code. For instance, while the Hadamard operator is transversal under the seven-qubit code, it is not so under the three qubit code. Note however that it is not a requirement that the logical encoding of a gate be equal to multiple single-qubit copies of itself: in the nine-qubit code, $\underline{X} = Z^{\otimes 9}$ and $\underline{Z} = X^{\otimes 9}$. Since $\underline{X}$ and $\underline{Z}$ can each be implemented via local gates they are transversal.

The existence of a transversal implementation of a gate under a given code is sufficient for that gate to be able to be performed fault tolerantly. Unfortunately, no code allows all members of any universal gate set to be implemented transversally [27]. Thankfully a transversal implementation is not a necessary condition for the fault tolerant implementation of a logical gate, and there do exist codes—including the seven-qubit Steane code—that can be implemented fault tolerantly.

1.6 Other models of quantum computation

The circuit model is not the only model of quantum computation, though as is the case with the various models of classical computing, each of the models of quantum computation so far proposed is capable of simulating each of the others with at most a polynomial increase in the runtime. Two other prominent and well-studied models are measurement-based and quantum-walk quantum computation. As the focus of the current work, the quantum-walk models are discussed in greater detail in subsequent chapters.

1.6.1 Measurement-based quantum computation

In the circuit model, the fiducial input is taken to be a product state. Entanglement is generated throughout the execution of the circuit, in accordance with the algorithm being implemented. Under the measurement-based or one-way model of quantum computation the initial state is instead a highly entangled state referred to as a 'resource state'. One such resource is the cluster state [28], in which an array of two-level qubit-like systems on the vertices of a square grid are each maximally entangled with their nearest neighbours. The initial state is independent of the algorithm to be implemented, and its entanglement is 'consumed' as single-qubit projective measurements are made on its two-level constituents.

Since measurement outcomes are probabilistic, the results of one round of measurements inform the choice of subsequent measurements. Starting at the left-hand side of the array, specific elements of the cluster state and measurement bases are prescribed based on the desired gate to be implemented. Local measurements on the entangled state transform the state of the rest of the cluster in such a way as to propagate an encoded n-qubit state across the cluster to the right, transforming it according to a chosen quantum algorithm along the way. In this manner, the cluster state and local measurements together are universal for quantum computation, but just as there exist many universal sets of gates in the circuit model the cluster state is not unique in its status as a resource for measurement-based quantum computing [29, 30].

Chapter 2

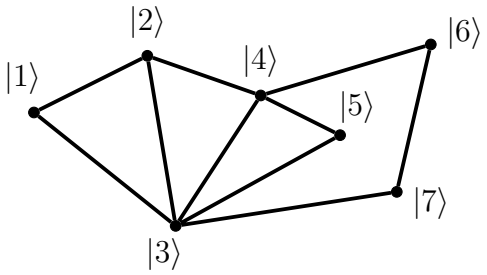
Quantum walks

Quantum walks are the quantum-mechanical analogue of classical random walks. At their simplest, these walks are the evolution of systems that inhabit and evolve on graphs, which in this context are generalized lattice-like structures composed of vertices and edges among them. For a given walk the graph encodes both the states the walker may occupy, and the allowed transitions among them. A simple illustrative example can be found in Figure 2.1. Due to the prominent nature of graphs within the realm of quantum walks, a cursory introduction to graph theory precedes a review of quantum-walk systems.

2.1 Graph theory

A comprehensive discussion of graph theory is beyond the scope of this work, but as repeated use of certain key concepts is made throughout it, a review of the relevant notations and definitions is presented here. More thorough treatments can be found, for example, in the textbooks by Biggs [31], Godsil and Royle [32], and Agnarsson and Greenlaw [33].

A *graph* $G = (V, E)$ is defined by a set V of vertices and a set $E \subseteq V \times V$ of edges between pairs of vertices. A *directed* graph allows for an edge to connect vertex u to vertex v, without also connecting v to u. Only *undirected* graphs, in which there is an edge from u to v if and only if there is an edge from v to u, are considered here. An edge "between" a vertex and itself is a *self-loop*. A *weighted* graph $G = (V, E, w)$ is a graph augmented by a weighting function $w : E \to \mathbb{R}^+$ that assigns to each edge a positive real number. A graph with no such function is called *unweighted*. Edge weights



(a)

Quantum walker:

$$|\psi\rangle = \sum_{v=1}^{7} \alpha_v |v\rangle$$

Vertex basis state:

$$|v\rangle \doteq \begin{pmatrix} 0 & \cdots & 1 & \cdots & 0 \end{pmatrix}^T$$
$$\uparrow v\text{th entry}$$

(b)

Figure 2.1: Illustrative example of a graph–quantum-walk system. (a) A graph contains vertices labelled by states $|v_i\rangle$. (b) The state of a quantum walker on the graph is a normalized superposition of the vertex basis states, which are represented canonically as the unit vectors $|v_i\rangle$ with the ith entry equal to 1.

can be thought of as transition rates, or as inversely proportional to a distance between vertices (though actual distances in graphical representations of graphs generally carry no meaning). If all weights in a given graph are integers, then the graph can also be thought of as having *multiple edges*, meaning two or more unweighted edges between a single pair of vertices. A *simple* graph is an unweighted graph with no self-loops. A *connected* graph is one in which there is a sequence of edges joining any two vertices. That is, given any pair of distinct vertices $\{u, v\} \in V$, there exists a sequence $(e_1, \ldots, e_l)$ of edges $e_i = (u_i, v_i) \in E$ connecting them, with $u_1 = u$, $v_l = v$, and $u_i = v_{i-1}$ for each $i \in \{2, \ldots, l\}$. Such a sequence is called a *path*; if $u = v$ and the sequence is non-empty, then it is called a *cycle*. A *simple path* is a path with no repeated vertices, and a cycle with no repeated vertices other than the required repetition of the first vertex as the last is a *simple cycle*.

A particularly useful description of a graph G is that of its *adjacency matrix*, defined for an n-vertex graph as the $n \times n$ matrix $A(G) = \{a_{uv}\}$, with a_{uv} equal to the weight of the edge between vertices u and v. The adjacency matrix is thus written in the *vertex basis*, spanned by the orthonormal vectors $\{|v\rangle\}_{v=1}^{N}$ where each $|v\rangle \in \mathbb{R}^N$ is the unit

vector corresponding to the vth dimension as for example in Figure 2.1(b). When the graph in question is clear from context, the adjacency matrix can also be written simply as A, without the explicit dependence on G. For an unweighted graph all entries of the adjacency matrix are either 0 or 1, and the adjacency matrix of an undirected graph is symmetric. Finally, the *degree* of a vertex in an unweighted graph, possibly with multiple edges, is the number of edges attached to the vertex; a graph is *regular* if all of its vertices have the same degree. It is straightforward to see that the degree of a vertex in an undirected, unweighted graph can be calculated as the sum of the entries in the row (or column) corresponding to that vertex in the graph's adjacency matrix.

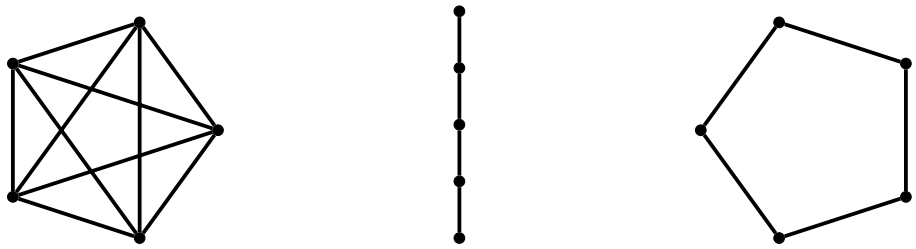
2.1.1 Examples of common graphs and graph classifications

The *complete graph* on N vertices, denoted K_N, has an edge between every pair of vertices. With J_N the all-ones $N \times N$ matrix, the adjacency matrix of a complete graph is

$$A(K_N) = J_N - I_N = \sum_{u \neq v} |u\rangle\langle v| = \begin{pmatrix} 0 & 1 & 1 & & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & & 1 \\ & \vdots & & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix}. \tag{2.1}$$

For example, K_5 can be seen in Figure 2.2(a).

Two graphs closely related to each other are the *path* and *cycle* on N vertices, P_N and C_N respectively. The path has two 'end' vertices of degree 1, and $N-2$ degree-2 vertices connected sequentially between the ends. The cycle is the same graph, with an additional edge joining the ends directly to each other. The examples P_5 and C_5 are shown in Figures 2.2(b) and (c), respectively. The adjacency matrix of the path is tridiagonal, with ones on the first upper and lower diagonals, and zeroes on the main diagonal and in every other entry; the adjacency matrix of the cycle contains two additional ones, in



(a) K_5 (b) P_5 (c) C_5

Figure 2.2: The complete, path, and cycle graphs on five vertices. (a) In the complete graph K_5, each vertex is connected to every other and thus has degree 4. (b) The path P_5 can be turned into (c) the cycle C_5 by adding an edge connecting the top vertex to the bottom.

the upper-right and lower-left corners:

$$A(P_N) = \sum_{v=1}^{\mathbb{N}-1} |v+1\rangle\langle v| + \text{H.c.} = \begin{pmatrix} 0 & 1 & 0 & & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & & 0 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}, \tag{2.2a}$$

and with vertices labelled modulo N so that $N+1=1$,

$$A(C_N) = \sum_{v=1}^{\mathbb{N}} |v+1\rangle\langle v| + \text{H.c.} = \begin{pmatrix} 0 & 1 & 0 & & 0 & 1 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & & 0 & 1 \\ 1 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{2.2b}$$

Note that the definitions of K_2 and P_2 lead to the same graph of two vertices with a single edge between them. Likewise K_3 and C_3 both describe a graph containing three vertices connected by three edges so as to form a triangle. It is said that K_2 is said to be isomorphic to P_2, and K_3 to C_3. In general if the adjacency matrices of two graphs are equal up to a permutation of the vertex basis, then the graphs are isomorphic; a more

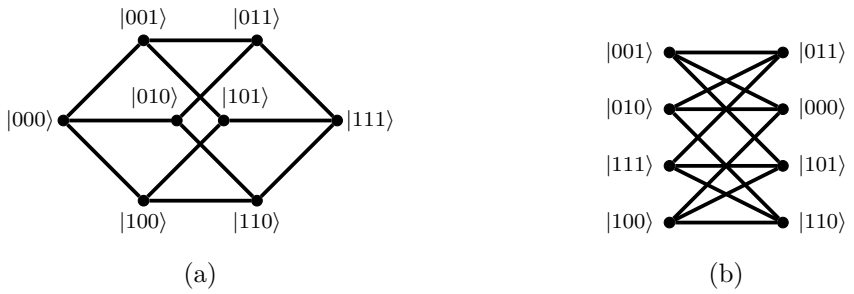


Figure 2.3: (a) The 3-hypercube is simply a cube. (b) By rearranging the vertices of the 3-hypercube into two columns, it can be seen that the graph is bipartite.

precise definition is given here.

Definition 2.1. *An* isomorphism *between two graphs* $G_i = (V_i, E_i, w_i)$ *for* $i \in \{1, 2\}$ *is a bijection* $f : V_1 \to V_2$ *such that* $(u_1, v_1) \in E_1 \Leftrightarrow (f(u_1), f(v_1)) \in E_2$, *and* $\forall (u_1, v_1) \in E_1$, $w_1(u_1, v_1) = w_2(f(u_1), f(v_1))$.

The problem of determining whether two graphs are isomorphic is in NP, although for the vast majority of arbitrarily chosen pairs of graphs non-isomorphism can be established efficiently. For example, if the graphs have different numbers of vertices then they cannot be isomorphic. A related concept that arises if G_1 and G_2 are known to be the same graph, i.e. when $G_1 = G = G_2$, is that of a graph automorphism.

Definition 2.2. *An* automorphism *of a graph* $G = (V, E, w)$ *is a permutation* $\sigma : V \to V$ *such that* $(u, v) \in E \Leftrightarrow (\sigma(u), \sigma(v)) \in E$ *and* $\forall (u, v) \in E$, $w(u, v) = w(\sigma(u), \sigma(v))$.

Another example of a well-studied family of graphs is give by the n-hypercubes, or simply n-cubes, on $N = 2^n$ vertices. For a specific n the vertices are labelled by the n-bit strings representing the integers from $0 = 00 \cdots 0_2$ to $N - 1 = 11 \cdots 1_2$, and there is an edge between every pair of vertices whose labels differ on exactly one bit. Since for every bitstring of length n there are exactly n bistrings that differ in exactly one bit, the

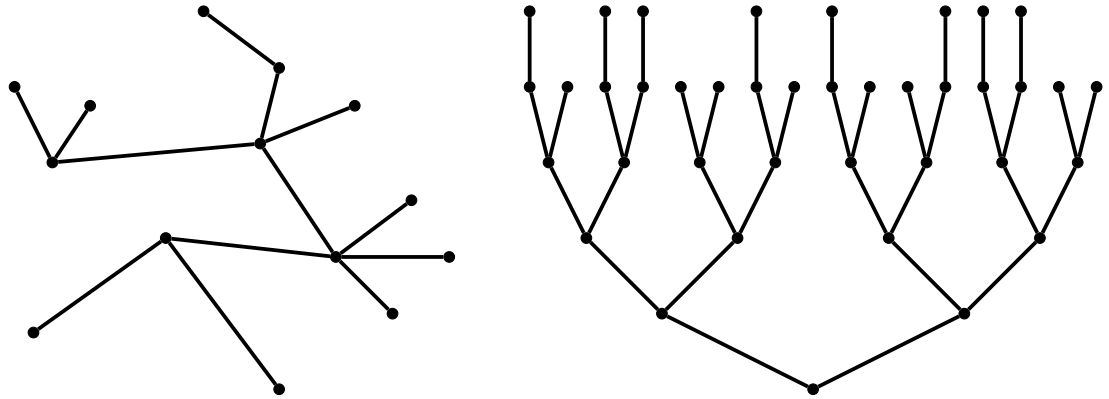


Figure 2.4: Two tree graphs. Both are connected and cycle-free, but (a) is unrooted and unbalanced while (b) is a balanced binary tree of depth 5.

n-cube is regular with vertex degree n. For each vertex there is a unique vertex whose label differs from that of the first in every bit; such pairs are called *antipodes*, and on any hypercube there is an automorphism that relabels a given pair of antipodal vertices as $|00\cdots0\rangle$ and $|11\cdots1\rangle$. The 2-cube is simply a square, isomorphic to C_4, and the 1-cube is the path on two vertices, isomorphic to both P_2 and K_2. The hypercube also provides an example of a *bipartite* graph, in which the vertices can be partitioned into two sets such that every edge of the graph connects a vertex in one of the sets to a vertex in the other. Figure 2.3 depicts two representations of the 3-hypercube, one in which it appears as a simple cube, and one in which its bipartite nature is manifest.

Finally, a *tree* is a graph in which there is a unique simple path between any two vertices. That is, a connected graph that contains no cycles is a tree. The trees that appear in the context of walk-based algorithms typically have one vertex singled out as the *root* of the tree, which provides a natural arrangement of the vertices in layers according to their distances to the root, in which case the parent of a vertex is its unique neighbour that is closer to the root, while its children are the rest of its neighbours, which are further from the root. Any vertex (other than the root) that has degree 1 is called a

leaf. A d-ary tree is a rooted tree that has no vertices with more than d children, and a tree is *balanced* if the distances from the root to each leaf differ by at most 1. Figure 2.4 contains two examples illustrating these features.

2.2 Classical random walks

A simple introduction to random walks is provided by the one-dimensional discrete-time random walk on the line, or the infinite path graph $P_{\aleph_0}$. The walker begins at the origin, at position $x = 0$, and proceeds to take steps by flipping a coin and taking a pace to the right, or incrementing its position on the line, if the coin comes up 'heads', and to the left on 'tails'. For simplicity, assume for now that the coin is fair. Since the walker beings with certainty on the origin, the probability distribution of its position after zero steps (i.e. before the first step) is given by $p_0(0) = 1$. After one flip of the coin there is a fifty per cent chance of the walker's having moved to each of $x = \pm 1$, leading to the probabilities $\{p_1(-1) = \frac{1}{2}, p_1(1) = \frac{1}{2}\}$. On the second step there are two paths the walker can have taken that return it to the origin, and one path to arrive at each of $x = \pm 2$; the probability distribution is given by $\{p_2(-2) = \frac{1}{4}, p_2(0) = \frac{1}{2}, p_2(2) = \frac{1}{4}\}$.

Continuing in this manner, moving half of the probability on a site to each of its neighbours for every step leads to the probability distributions in Table 2.1. The numerators form Pascal's triangle, padded with alternating empty sites, and the denominators after s steps are 2^s. To investigate the expected behaviour of this walk on the line, note that the coin flips can be modelled as a sequence of random variables, $(C_1, C_2, \ldots)$, with each C_i returning $+1$ if flip i is heads, and -1 if tails. The position of the walker after s steps is therefore a random variable P given by

$$P(s) = \sum_{i=1}^{s} C_i. \tag{2.3}$$

Steps, s	$x=$	-5	-4	-3	-2	-1	0	$+1$	$+2$	$+3$	$+4$	$+5$
0							1					
1						$\frac{1}{2}$		$\frac{1}{2}$				
2					$\frac{1}{4}$		$\frac{2}{4}$		$\frac{1}{4}$			
3				$\frac{1}{8}$		$\frac{3}{8}$		$\frac{3}{8}$		$\frac{1}{8}$		
4			$\frac{1}{16}$		$\frac{4}{16}$		$\frac{6}{16}$		$\frac{4}{16}$		$\frac{1}{16}$	
5		$\frac{1}{32}$		$\frac{5}{32}$		$\frac{10}{32}$		$\frac{10}{32}$		$\frac{5}{32}$		$\frac{1}{32}$

Table 2.1: Probability distribution over the first five steps of the discrete-time random walk on the line with a fair coin.

Since the coin is fair, the expectation of each flip is $\langle C_i \rangle = 0$, and so the expected position of the walker is $\langle P \rangle = 0$ as well. It turns out that the probability distribution of P approaches a Gaussian in the large-s limit, and while the origin is the position with the single highest probability to find the walker, there is a significantly greater probability to find it *not* at the origin.

This is captured by looking at the expected distance of the walker from the origin, the square of which after s steps is

$$D^2(s) = \left| \sum_{i=1}^{s} C_i \right|^2 = \left(\sum_{i=1}^{s} C_i \right)^2 . \tag{2.4}$$

The expansion of this expression contains s^2 terms. Of these, s are of the form C_i^2 while the remainder are $C_i C_j$ for some $i \neq j$. The expectation value of D^2 after s steps is therefore

$$\langle D^2(s) \rangle = \sum_{i=1}^{s} \langle C_i^2 \rangle + \sum_{i \neq j} \langle C_i C_j \rangle, \tag{2.5}$$

in which the second sum vanishes because half of its summands equal $+1$ when the flips i and j are the same, and half yield -1 when they are different. The first sum on the other hand yields s because $(+1)^2 = (-1)^2 = 1$. Therefore, the expected distance of the walker from the origin after s steps is $\sqrt{\langle D^2(s) \rangle} = \sqrt{s}$. The probability distribution of the classical discrete-time random walk on the line is demonstrated in Figure 2.5(a).

The simplest generalization to this random walk is to make the coin unfair, yielding heads with probability p and tails with probability $1-p$. Another extension is to expand the line to additional dimensions, such as a two-dimensional grid or three-dimensional lattice, or more generally to arbitrary graphs. The walk is then defined by the underlying graph and at each vertex a probability distribution over the attached edges. However, the straightforward walk on a line with a fair coin will prove sufficient to demonstrate the stark contrasts between quantum and classical walks.

The probability distribution for the discrete-time random walk on the line arises due to the choice of initial state, the weighting of the coin—in this case equal between the two options—and the structure of the graph on which the walker moves. Given the adjacency matrix A for the infinite path, an initial vector $\vec{p}^{(0)} = \delta_{u0}$ of probabilities over the integers such that $p_u^{(0)}$ is the probability to find the walker on vertex u initially, and a fair coin, the probability distribution after s steps is given be

$$\vec{p}^{(s)} = \left(\frac{1}{2}A\right)^s \vec{p}^{(0)} \doteq S^s \vec{p}^{(0)}. \tag{2.6}$$

This definition of a step operator S can be applied more generally, as a matrix with entries S_{uv} equal to the probability for a walker on vertex v to step to u on a given graph. An immediate consequence of this definition is that $0 \leq S_{uv} \leq 1$ for each entry of S, and for every vertex v,

$$\sum_u S_{uv} \stackrel{!}{=} 1 \tag{2.7}$$

in order to conserve probability. This procedure of multiplying the edge weights in an adjacency matrix by the probability for a walker to step along each edge and acting the resulting operator s times on an initial probability distribution can be applied to arbitrary graphs and initial positions of the walker. A similar scheme will be employed for the quantum case in Section 2.3, and the walk operator is also useful in defining the

continuous-time random walk.

2.2.1 Continuous-time random walks

The continuous-time version of a classical random walk is described by using a sequence of random variables to define the times at which steps are made, as well as one to determine which steps can be made. That is, a sequence of random variables $(T_1, T_2, \ldots)$ drawn from a distribution of strictly positive real numbers describes the waiting times between steps, such that step s occurs at time

$$t_s = \sum_{i=1}^{s} T_i. \tag{2.8}$$

The distribution from which the times are drawn defines a mean transition rate γ, such that for long walks it is expected that an average of γ steps are taken per unit time. If at some time t the probability distribution describing the location of the walk is given by $\vec{p}(t)$ then at a later time $t+h$, for some small but positive h, there is a probability γh that a step has been taken and the step operation S has been applied to the probability distribution, and a probability $1-\gamma h$ that the distribution is unchanged. That is,

$$\vec{p}(t+h) = (1-\gamma h)\vec{p}(t) + \gamma h\, S\vec{p}(t), \tag{2.9}$$

which for strictly positive h can be rewritten as

$$\frac{\vec{p}(t+h) - \vec{p}(t)}{h} = \gamma(S-I)\vec{p}(t). \tag{2.10}$$

Taking the limit as $h \to 0^+$ yields the differential equation $\partial_t \vec{p}(t) = \gamma(S-I)\vec{p}(t)$, which has the solution

$$\vec{p}(t) = \mathrm{e}^{\gamma t(S-I)}\vec{p}(0). \tag{2.11}$$

This is the evolution equation for the continuous-time random walk.

Note that probability is conserved by this equation, in that if $\sum_v p_v(0) = 1$, then $\sum_v p_v(t) = 1$ for all times $t > 0$. This is due to the fact the rows of S each sum to unity, so that

$$\sum_v p_v(t) = \sum_v \left(\mathrm{e}^{\gamma t(S-I)}\vec{p}(0)\right)_v = \sum_v \left[\vec{p}(0) + \left(\sum_{n=1}^{\infty} \frac{(\gamma t)^n}{n!}(S-I)^n\right)\vec{p}(0)\right]_v . \tag{2.12}$$

By assumption the first term yields $\sum_v p_v(0) = 1$, and since $\sum_u S_{uv} = 1$ for each v,

$$\sum_v \left[(S-I)\vec{p}(0)\right]_v = \sum_v \sum_u (S-I)_{uv} p_v(0) = \sum_v p_v(0) \left[\sum_u (S-I)_{uv}\right] = 0. \tag{2.13}$$

Combining these two equations leads to the conclusion that $\sum_v p_v(t) = 1$, i.e. that the total probability is equal to unity for all time.

The limiting behaviour of the continuous-time random walk on the line is the same as that of its discrete-time counterpart. To see this, note that for large s,

$$\mathrm{e}^{\gamma t(S-I)} \approx \left(I + \frac{\gamma t}{s}(S-I)\right)^s . \tag{2.14}$$

Thus for long times $t = s/\gamma$, $s \gg \gamma$, the evolution of the continuous-time random walk is governed by

$$\mathrm{e}^{s(S-I)} \approx S^s, \tag{2.15}$$

which governs the discrete-time random walk, as in Equation (2.6).

Figures 2.5(a) and (b) depict the evolution of the probability distribution for the discrete- and continuous-time random walks respectively, as well as the spreading of the expected distance from the origin. In each case, one example random walk is superimposed on the distribution. The remainder of Figure 2.5 shows the evolution of comparable

quantum walkers, which exhibit linear $O(t)$ spreading in sharp contrast with the $O(\sqrt{t})$-width spread of the classical case. These results are elaborated upon in the next section.

2.3 Defining quantum walks

As with the classical random walk, there exist both discrete- and continuous-time versions of the quantum walk. The primary focus of the work presented in this thesis is on the continuous-time form of the quantum walk. The discrete-time model however was historically the first to be described [34], and is analogous to the perhaps more intuitive discrete-time classical random walk, so it is discussed first to provide an introduction to quantum walks in general. In the remainder of this section, those definitions and aspects of quantum walks that are salient to the current work are introduced. More in-depth reviews can be found, for example, in References [35, 36].

2.3.1 Discrete-time quantum walks

As in the classical case, the quantum walker is defined as as a system with a particular state space and a mechanism for transitioning between states in a set of discrete steps. However while the classical random walker moves randomly to a position that is then known with certainty, the probability amplitude of the quantum walker moves deterministically in a superposition that makes its position uncertain. Specifically, a discrete-time quantum walk takes place in a position space $\mathscr{H}_p$, augmented with a coin degree of freedom living in a coin space $\mathscr{H}_c$, so that the Hilbert space of the quantum walk is $\mathscr{H} = \mathscr{H}_p \otimes \mathscr{H}_c$. An orthonormal set of states associated with the vertices of a graph yield a basis for the position space of a walk on that graph. As for the classical random walk, it is instructive to begin with the straightforward example of a walk on a line, which already provides an example of the significant differences between the

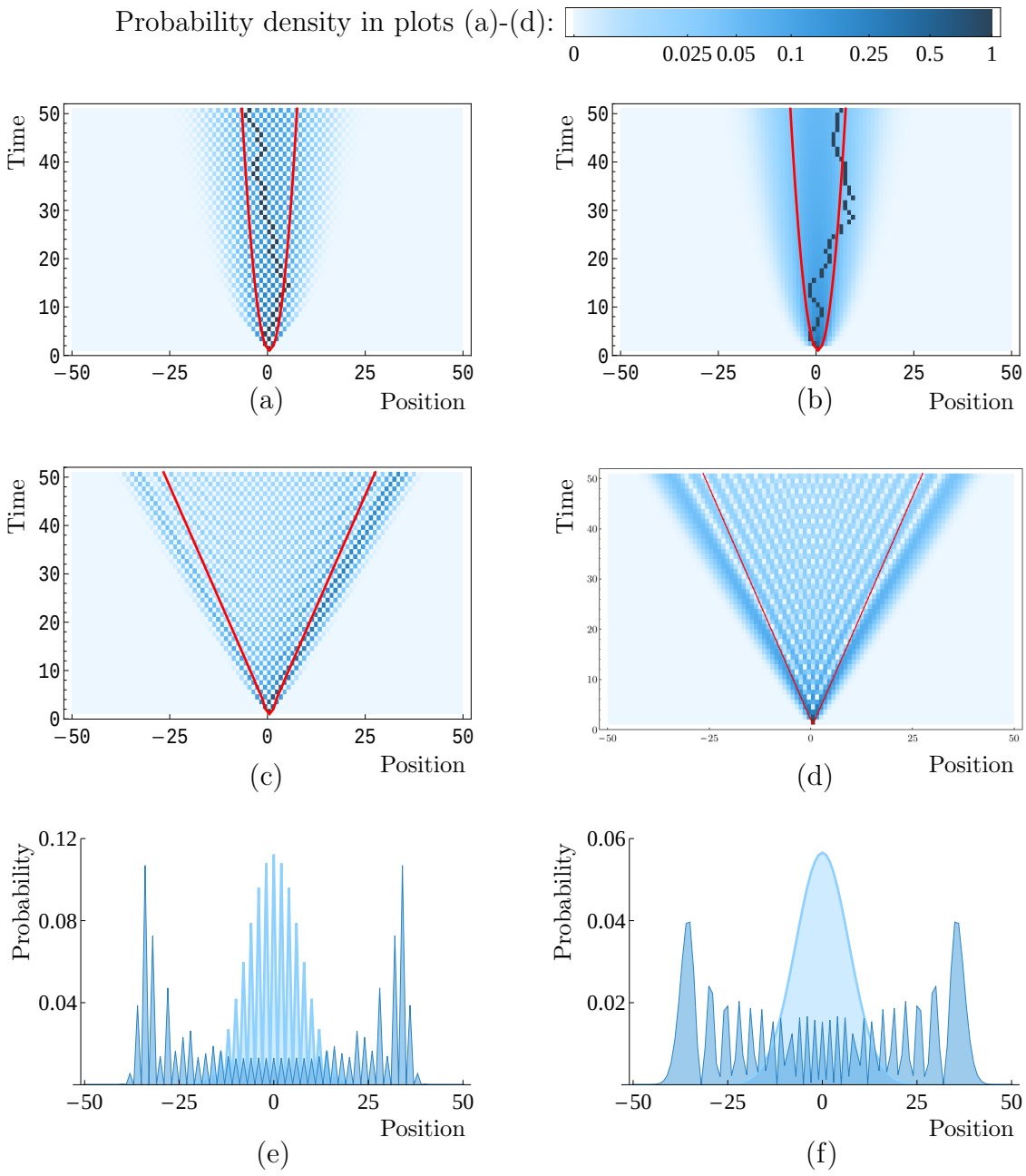


Figure 2.5: Comparison of the evolutions of four models of walk on the line. The density plots (a)-(d) show the probability distribution of the (a) discrete-time random, (b) continuous-time random, (c) discrete-time quantum, and (d) continuous-time quantum walks. Each begins with certainty at the origin at time $t = 0$. The superimposed red lines depict the expected distance from the origin in each case. The classical probability distributions also show an example of one specific walk each. Plots (e) and (f) contrast the probability distributions of the classical (light) and quantum (dark) walks in the discrete- and continuous-time cases, respectively, at time $t = 50$.

behaviours of quantum and classical walks.

Consider the position space $\mathscr{H}_p = \text{span}\{|v\rangle : v \in \mathbb{Z}\}$ and two-dimensional coin Hilbert space $\mathscr{H}_c = \text{span}\{|R\rangle, |L\rangle\} = \mathbb{C}^2$. The first difference encountered in the discrete-time quantum walk relative to a classical random walk is that the initial state of the coin must be specified, as well as that of the walker. For example, in the basis $|R\rangle = (1 \;\; 0)^T$ and $|L\rangle = (0 \;\; 1)^T$, consider the initial state

$$|\psi(0)\rangle \doteq |0\rangle \otimes \frac{1}{\sqrt{2}} \left(|R\rangle + \mathrm{i}|L\rangle\right) \tag{2.16}$$

and define the coin operation to be the Hadamard operator,

$$C \doteq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.17}$$

which 'flips' the coin subspace by transforming each of the basis elements into an equal superposition of the two of them. Once the coin has been flipped, a shift operator moves the walker in the direction of the coin state; explicitly, define

$$S \doteq \sum_{x=-\infty}^{\infty} \left(|x+1\rangle\langle x| \otimes |R\rangle\langle R| + |x-1\rangle\langle x| \otimes |L\rangle\langle L|\right), \tag{2.18}$$

which is unitary since $S^\dagger S = SS^\dagger = I_p \otimes I_c = I$, where I_p and I_c are the identity operators of the position and coin spaces respectively. A single step of the quantum walk consists of the application of a coin flip, followed by a shift. The shift operator acts on the entire Hilbert space $\mathscr{H}$, but the coin affects only its subspace $\mathscr{H}_c$, at least as commonly presented and as introduced in Equation (2.17); technically then one step of the walk is accomplished by the operator $S(I_p \otimes C)$. For notational brevity however, it is common practice to define the *walk operator* $U \doteq SC$, where the implicit inclusion of

the appropriate identity is understood. The state of the walker after s steps is given by

$$|\psi(s)\rangle \doteq U^s|\psi_0\rangle, \tag{2.19}$$

from which the probability for the walker to be found on vertex $|v\rangle$ can be found as $|\langle v|\psi(s)\rangle|^2$. This probability distribution over a finite subset of the line, with initial state given by Equation (2.16) and the Hadamard coin of Equation (2.17), is shown in Figure 2.5(c), and contrasted with the probability distribution of the classical discrete-time random walk in Figure 2.5(d).

As in the classical case the expected position of the walker after s steps,

$$\langle P(s)\rangle \doteq \langle\psi(s)|\left(\sum_{v=-\infty}^{\infty} v|v\rangle\langle v|\right)|\psi(s)\rangle, \tag{2.20}$$

vanishes in this example. Interestingly however, this is not the case for all fair coins. There exist coins that are fair, in the sense that they map each coin state to an equal superposition of the two, yet which due to different phases appearing in the initial superposition of the coin state, the coin itself, or both, the expected position of the walker shifts from the origin over time. The feature of note in the present case however, is that even with a fair coin that maintains $\langle P(s)\rangle = 0$ for all s, the expected distance from the origin increases quadratically more quickly than the same quantity in a comparable classical walk. For the quantum walk in question,

$$\sqrt{\langle D^2(s)\rangle} \doteq \left[\langle\psi(s)|\left(\sum_{v=-\infty}^{\infty} v^2|v\rangle\langle v|\right)|\psi(s)\rangle\right]^{\frac{1}{2}} \propto s, \tag{2.21}$$

as compared to the result of $\sqrt{s}$ classically. This quadratic speed-up can be attributed to interference phenomena. When a classical walker has the possibility to arrive on a given vertex after having followed one of many paths, the probability to find it there is the sum

of the probabilities to have taken each of the paths. For a quantum walk on the other hand, it is the probability amplitudes that add. Since these can be negative, sometimes the combinations of paths leading to a given vertex interfere destructively, decreasing or eliminating the probability to find the walker there.

A discrete-time quantum walk on a more general graph follows the prescription outlined here, of putting the coin into a superposition of directions at each vertex, shifting the walker along the edges of the graph, and repeating. If the graph is not regular then multiple coins are required, at least one for each degree present in the graph, and possibly as many as one for each vertex.

2.3.2 Continuous-time quantum walks

A simple observation suffices to begin a discussion of the continuous-time quantum walk, first introduced by Farhi and Gutmann [37]: the adjacency matrix of an undirected graph with real edge weights is real and symmetric, and therefore able to describe the (Hermitian) Hamiltonian of a physical system. Given an undirected graph $G = (V, E, w)$ with adjacency matrix

$$A_G \doteq \sum_{(u,v)\in E} w_{uv}|u\rangle\langle v|, \tag{2.22}$$

define the Hamiltonian of a quantum walker on G to be $\mathcal{H}_G \doteq -A_G$, so that given an initial normalized state

$$|\psi_0\rangle = \sum_{v\in V} \alpha_v|v\rangle, \tag{2.23}$$

the state of the walk at time $t > 0$ is given by the usual Schrödinger evolution,

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}\mathcal{H}_G t}|\psi_0\rangle = \mathrm{e}^{\mathrm{i}A_G t}|\psi_0\rangle. \tag{2.24}$$

Note that since the topology of the graph itself generates the time translations of the walker, no coin degree of freedom is necessary. In particular, this means that a continuous-time quantum walker requires no internal degrees of freedom.

Consider again the walk on the line for illustrative purposes. The required adjacency matrix is

$$A = \sum_{v=-\infty}^{\infty} |v+1\rangle\langle v| + \text{H.c.} \tag{2.25}$$

and the initial state is taken once more to be $|\psi_0\rangle = |0\rangle$. The Hamiltonian has energy eigenvalues $E_k = -2\cos(k)$ with associated non-normalizable eigenstates $|k\rangle$ of the form

$$\langle v\,|k\rangle = \mathrm{e}^{\mathrm{i}kv} \tag{2.26}$$

with $k \in (-\pi, \pi]$. The initial state can be expressed in the energy eigenbasis as

$$|0\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{d}k\,|k\rangle\langle k|0\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{d}k\,|k\rangle, \tag{2.27}$$

from which it is seen that the state of the walker evolves as

$$|\psi(t)\rangle \doteq \mathrm{e}^{-\mathrm{i}\mathcal{H}t}|0\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{d}k\,\mathrm{e}^{2\mathrm{i}\cos(k)t}|k\rangle. \tag{2.28}$$

Insertion of the identity, resolved in terms of the vertex basis as $\sum_v |v\rangle\langle v|$, determines the evolution of the probability amplitude on each vertex of the graph,

$$|\psi(t)\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{d}k\,\mathrm{e}^{2\mathrm{i}\cos(k)t}\sum_{v=-\infty}^{\infty}|v\rangle\langle v|k\rangle = \sum_{v=-\infty}^{\infty}\left(\frac{1}{2\pi}\int_{-\pi}^{\pi}\mathrm{d}k\,\mathrm{e}^{2\mathrm{i}\cos(k)t}\mathrm{e}^{\mathrm{i}kv}\right)|v\rangle. \tag{2.29}$$

This analysis allows the probability distribution of the location of the walker on a finite subset of the infinite line to be plotted at arbitrary times. Such plots can be seen in Figure 2.5, showing the evolution of $|\langle v\,|\psi(t)\rangle|^2$ over time.

The absence of a coin space in the continuous quantum walk and dependence upon one in the discrete case imply that the two models are not isomorphic in any limit. Nevertheless the behaviours of the two walks on the line are qualitatively very similar, as can be seen by comparing Figure 2.5(c) to (d), and (e) to (f). Despite the lack of a direct isomorphism between the two models, there do exist limits in which their behaviours can be shown to be not merely qualitatively similar, but quantitatively equivalent [38, 39].

2.4 Periodicity and perfect state transfer

Classical random walks have, by definition, random evolution. It is therefore impossible to know which vertex it will occupy after any number of steps. In particular, if a walk begins on one vertex there is no way to determine in advance if or when it will arrive on a specified second vertex, or return to the first. Quantum walks on the other hand evolve unitarily under Schrödinger's wave equation for discrete systems, and just as a the ripples propagating away from a pebble dropped into the centre of a circular pond will interfere constructively to reconverge at the centre, a quantum walker on a graph can undergo constructive interference such that it returns with certainty to its starting vertex. Similarly, there exist graphs on which the state of a walker originating on one vertex will transfer with unit probability—that is, perfectly—to a second vertex. This phenomenon is called *perfect state transfer* (PST) [40–43], and appears repeatedly, along with generalizations of it, in later chapters.

The simplest example of perfect state transfer is provided by a walker initialized on one of the two vertices of P_2, the path graph on two vertices, shown in Figure 2.6(a). The adjacency matrix of this graph is equal to the Pauli-X operator,

$$A_{K_2} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = X, \tag{2.30}$$

and a walker initialized on vertex $|1\rangle$ will evolve according to

$$|\psi(t)\rangle = \exp\left(\mathrm{i}A_{K_2}t\right)|1\rangle = \cos(t)|1\rangle + \mathrm{i}\sin(t)|2\rangle. \tag{2.31}$$

In general there is a non-zero probability to find the walker on either of the two vertices, but for times $t_{1,n} \doteq n\pi$ and $t_{2,n} \doteq (2n+1)\frac{\pi}{2}$, $n \in \mathbb{Z}$,

$$\langle 1|\psi(t_{1,n})\rangle = \langle 2|\psi(t_{2,n})\rangle = 1, \tag{2.32}$$

and the location of the walker is known with certainty. Thus there is perfect state transfer from vertex $|1\rangle$ to vertex $|2\rangle$ in time $\pi/2$, and the graph is *periodic* with period π. Note that it is possible for a vertex or a subset of vertices within a graph to be periodic, while other vertices are not. In the case of P_2 however, both vertices have the same period and so the graph as a whole is said to be periodic.

The conditions dictating whether or not a graph exhibits perfect state transfer or is periodic are related to the spectrum of its adjacency matrix. If a graph has two vertices $|u\rangle$ and $|v\rangle$ that are equivalent up to an automorphism, then it cannot exhibit PST between them unless the ratios of differences of pairs of distinct eigenvalues are rational [42]. Furthermore, if there is PST between vertices $|u\rangle$ and $|v\rangle$ in time t_p then each of $|u\rangle$ and $|v\rangle$ is periodic in time $2t_p$ [44]. To see the importance of the spectrum of a graph to perfect state transfer, consider first graph periodicity. If a graph with adjacency matrix A is periodic, then there exists some time $t_p > 0$ at which the resulting unitary evolution operator is proportional to the identity. That is, for some phase $\theta \in [0, 2\pi)$ it must be the case that

$$U(t_p) \doteq \mathrm{e}^{\mathrm{i}At_p} \stackrel{!}{=} \mathrm{e}^{\mathrm{i}\theta}I. \tag{2.33}$$

At the same time, this operator yields a spectral decomposition in terms of its eigenvalues

λ_i and corresponding eigenvectors $|\phi_i\rangle$,

$$U(t) \doteq \mathrm{e}^{\mathrm{i}At} = \sum_i \mathrm{e}^{\mathrm{i}\lambda_i t}|\phi_i\rangle\langle\phi_i|. \tag{2.34}$$

Since the $|\phi_i\rangle$ form an orthonormal basis, this sum resolves to the identity if there exists a time at which the phase factors can be factored out. Expanding on this line of reasoning, Godsil proves the following result.

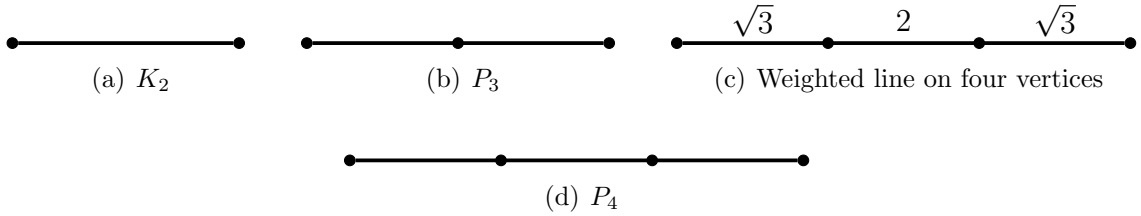
Theorem 2.1 (Cf. Theorem 3.1 in Reference [45])**.** *A graph G is periodic if and only if the ratio of any two of its non-zero eigenvalues is rational.*

When this rationality condition is satisfied it is straightforward to see that there exists a time t_p at which G is periodic. For each eigenvalue λ_i of the adjacency matrix A of G there exist integers p_i and q_i, $q_i \neq 0$, such that $\lambda_i = \lambda_1 p_i/q_i$. There then exists a common denominator q such that defining $\tilde{p}_i = qp_i/q_i$ leads to

$$\lambda_i = \lambda_1 \frac{\tilde{p}_i}{q}. \tag{2.35}$$

Therefore at time $t_p \doteq \pi q/\lambda_1$, the phases in Equation (2.34) become $\mathrm{i}\lambda_i t_p = \mathrm{i}\pi\tilde{p}_i$, and since the $\tilde{p}_i$ are integers, and $U(t_p) = -I$.

Return now to the specific case of perfect state transfer on the one-segment path P_2. If another vertex is added to the line, turning it into the graph of Figure 2.6(b), the path P_3 on three vertices, it can be shown that there is perfect state transfer from each end of the line to the other in time $t_p = \pi/\sqrt{2}$ and the middle vertex is periodic in the same time. The graph as a whole is thus periodic in twice the transfer time, with period $T = \sqrt{2}\pi$. One might conjecture that this trend will continue for path graphs of arbitrary length, but the next in the sequence, P_4, provides a counter-example. Indeed, for all $N > 3$, there is no perfect state transfer or periodicity on P_N. The graphs P_2 and



(a) K_2 (b) P_3 (c) Weighted line on four vertices

(d) P_4

Figure 2.6: Three graphs that exhibit perfect state transfer and one that doesn't. (a) K_2 has PST from each vertex to the other and (b) P_3 from each end to the other. Both graphs are periodic. (c) This weighted version of P_4 also exhibits PST and periodicity, however the unweighted (d) P_4 in contrast is not periodic, and does not exhibit PST between any pair of vertices.

P_3 have eigenvalues $\{\pm 1\}$ and $\{0, \pm\sqrt{2}\}$ respectively, but the eigenvalues $\{\pm\varphi, \pm\varphi^{-1}\}$ of P_4 (with φ the golden ratio) do not satisfy the criterion of Theorem 2.1.

However, if the line on four vertices is reweighted as in Figure 2.6(c), then it *is* periodic, and exhibits PST from each vertex $|v\rangle$ to its mirror image $|5-v\rangle$ ($v \in \{1, \ldots, 4\}$) in time $t_p = \pi/2$. It turns out that this trend continues, and a line of arbitrary length can be made to exhibit perfect state transfer by choosing appropriate weights for its edges. For a line on N vertices, the appropriate weight for the edge between vertices $|v\rangle$ and $|v+1\rangle$ is

$$w(v, v+1) = \sqrt{v(N-v)}. \tag{2.36}$$

The eigenvalues of the adjacency matrix are $\{-(N-1), -(N-3), \ldots, N-1\}$, which are all integral and therefore all pairwise ratios of non-zero eigenvalues are rational. Remarkably the time to cross the path is $t_p = \pi/2$, independent of its length. Since the eigenvalues are integers, one simply takes $\tilde{p}_i = \lambda_i$ and $q = \lambda_1$ to satisfy Equation (2.35). With $\lambda_1 = N-1$ the largest eigenvalue, the period of the graph becomes $t_p = \pi q/\lambda_1 = \pi$. In half the period, PST occurs between mirrored vertices.

This behaviour was first described by Christandl *et al.* in the context of quantum spin chains [41]. They consider a chain of spin-$\frac{1}{2}$ stationary particles with nearest-neighbour

interactions, and show that the evolution of the single-excitation subspace corresponds to a quantum walk on the line, undergoing perfect state transfer if the couplings of the spin chain are tuned to match the described edge weights.

The spin-chain model of quantum state transfer on graphs was introduced by Bose, who considered only uniform couplings along the length of the chain [40] and thus could obtain perfect transfer only across two or three spins. Given a set of N spins in a one-dimensional lattice with nearest-neighbour couplings, the (fixed) positions of the spins are readily identified with vertices and the couplings between them with edges in P_N. For example, with $\tau_{i,i+1} = \tau_{i+1,i}$ a tunable coupling constant, consider the simple XY coupling of the Hamiltonian

$$\mathcal{H}_{XY} = -\frac{1}{2}\sum_{i=1}^{N-1} \tau_{i,i+1}\left(X_i X_{i+1} + Y_i Y_{i+1}\right). \tag{2.37}$$

The Pauli spin operators on spin i are X_i, Y_i, and Z_i, and the Hamiltonian $\mathcal{H}_{XY}$ commutes with the total z component of the spin, $\sum_i Z_i$, which is therefore conserved. Thus the 2^N-dimensional Hilbert space of N spins decomposes into invariant subspaces, each with a constant total spin projection in the z direction. Of interest here are the one-dimensional zero-excitation subspace $\mathscr{H}_0$ spanned by the all-spin-down state $|\downarrow\downarrow\cdots\downarrow\rangle$, which will encode one of the computational basis states of a qubit, and the N-dimensional one-excitation subspace $\mathscr{H}_1$, which will encode the other and that is spanned by the states with a single up spin and $N-1$ down spins, $\{|\uparrow\downarrow\downarrow\cdots\downarrow\rangle, |\downarrow\uparrow\downarrow\cdots\downarrow\rangle, \ldots, |\downarrow\cdots\downarrow\uparrow\rangle\}$.

The evolution of a state in $\mathscr{H}_1$ can be mapped to a quantum walk on a weighted path on N vertices, with weights specified by Equation (2.36) and vertex $|v\rangle$ corresponding to the state in which spin v is up and the rest down. A single-qubit state $|\psi\rangle = \alpha|\underline{0}\rangle + \beta|\underline{1}\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$ can be transmitted from one end of the chain to the other as follows. (Here the computational states are underlined to distinguish them from vertex states;

this convention is used throughout the subsequent chapters.) The spin chain is cooled to its ground state, $|\downarrow\downarrow\cdots\downarrow\rangle$, which is a zero-energy eigenstate of $\mathcal{H}_{XY}$. The first spin in the chain is then rotated into the state $\alpha|\downarrow\rangle + \beta|\uparrow\rangle$, encoding $|\psi\rangle$, the state to be transferred. The state of the spin chain is thus

$$|\Psi(0)\rangle = \alpha|\downarrow\rangle^{\otimes n} + \beta|\uparrow\rangle \otimes |\downarrow\rangle^{\otimes(n-1)}, \tag{2.38}$$

which then evolves under $\mathcal{H}_{XY}$. The state $|\downarrow\rangle^{\otimes n}$ is stationary, while the state with coefficient β maps to a quantum walk on P_N initially on the first vertex, $|1\rangle$. With the coupling constants $\tau_{i,i+1}$ tuned to match the corresponding edge weights as given by Equation (2.36), the walker evolves in time $t_p = \pi/2$ to vertex $|N\rangle$, leaving the spin chain in the state

$$|\Psi(t_p)\rangle = \alpha|\downarrow\rangle^{\otimes n} + \beta|\downarrow\rangle^{\otimes(n-1)} \otimes |\uparrow\rangle. \tag{2.39}$$

The state of the final spin in the chain is $\alpha|\downarrow\rangle + \beta|\uparrow\rangle$, and the computational state initially encoded by the first spin has been perfectly transferred to the final spin.

This XY model further allows the dynamics of quantum walks on more general graphs. A quantum walk on an N-vertex graph $G = (V, E, w)$ can be mapped to the single-excitation subspace of a system of N spin-$\frac{1}{2}$ particles under a spin-preserving Hamiltonian of the form

$$\mathcal{H}_{XY}(G) = -\frac{1}{2} \sum_{(u,v)\in E} w(u,v) \left(X_u X_v + Y_u Y_v\right), \tag{2.40}$$

which is a straightforward generalization of Equation (2.37) to arbitrary weighted graphs.

2.5 Multiple walkers on a graph

In Chapter 6, multiple quantum walkers evolving and interacting on a single graph are used to significantly decrease the number of vertices required to simulate a given compu-

tation, as compared to the schemes presented in the intervening chapters. A convenient description of multiple walkers—in particular indistinguishable ones—on a single graph is provided by the second-quantized notation commonly used to describe many-particle quantum systems. A thorough treatise on the subject can be found, for example, in the textbook of Fetter and Walecka [46], but a brief introduction sufficient for the subsequent discussion is provided here.

In quantum mechanics, particles of a given type are generally indistinguishable. For example, given two protons there exist no experiments capable of determining that *this* is the first proton, and *that* is the second; there are simply two protons. This is not to say that all protons must be in the same state. A pair of protons can well exist in a state described as "one proton has energy E_1 and one proton has energy E_2", but unlike a classical system in which one could say, for example, that "the red ball is moving quickly and the green ball is stationary" there is no way quantum mechanically to colour the protons so as to specify which one has which energy. This experimental fact presents a problem to the traditional first-quantized notation of quantum mechanics, in which a two-proton state $|E_1\rangle \otimes |E_2\rangle$ is perfectly valid. The solution within first quantization begins with the introduction of the permutation operator P, which interchanges two particles. Interchanging the particles twice returns them to their initial state, $P^2|\psi\rangle\otimes|\phi\rangle = |\psi\rangle\otimes|\phi\rangle$, so

$$P|\psi\rangle \otimes |\phi\rangle = \pm|\phi\rangle \otimes |\psi\rangle. \tag{2.41}$$

Particles obeying $P|\psi\phi\rangle = +|\phi\psi\rangle$ are called *bosons*, while those that acquire a non-trivial phase under exchange, $P|\psi\phi\rangle = -|\phi\psi\rangle$, are *fermions*.[1]

A two-particle state that represents the statement "one particle is in state $|\psi\rangle$ and

[1] At least, these are the only possibilities in three (or more) dimensions. For particles confined to two dimensions, topological considerations allow for so-called *anyons*, which can acquire any phase factor $e^{i\theta}$ when they are interchanged.

another is in state $|\phi\rangle$" can be written

$$|\Psi_{12}\rangle = \frac{1}{\sqrt{2}} \left(|\psi\phi\rangle \pm |\phi\psi\rangle \right), \tag{2.42}$$

where the relative phase factor ± 1 depends on whether the particles are bosons or fermions. Constructing such a state is referred to as *(anti)symmetrization*, since the result is a multi-particle state that is either symmetric or antisymmetric under exchange. This procedure succeeds in creating multi-particle states that accurately represent the physically allowed states of collections of multiple particles, the notation quickly becomes unwieldy; the extension of Equation (2.42) to n particles contains $n!$ terms.

2.5.1 Second-quantized notation

An alternative method with which to describe multi-particle quantum states is called *second quantization*, simply because historically it was the second formalism developed for the description of quantized systems. States representing multiple particles are constructed with the use of 'creation' operators $c^\dagger$ that add another particle to a given state, and a zero-particle state called the vacuum and denoted either $|\text{vac}\rangle$ or $|0\rangle$. For example, the two-particle state of Equation (2.42) becomes

$$|\Psi_{12}\rangle = c_\psi^\dagger c_\phi^\dagger |\text{vac}\rangle. \tag{2.43}$$

The symmetry (or antisymmetry) of the state is determined by the commutation relations of the creation operators $c^\dagger$, and their Hermitian conjugates c, called annihilation operators.

For bosons two creation operators commute, as do two annihilation operators, while a creation operators commutes with annihilation operators except for that corresponding

to the same state:

$$[c_\psi^\dagger, c_\phi^\dagger] \doteq c_\psi^\dagger c_\phi^\dagger - c_\phi^\dagger c_\psi^\dagger = 0, \tag{2.44a}$$

$$[c_\psi, c_\phi] \doteq c_\psi c_\phi - c_\phi c_\psi = 0, \tag{2.44b}$$

$$[c_\psi, c_\phi^\dagger] \doteq c_\psi c_\phi^\dagger - c_\phi^\dagger c_\psi = \delta_{\psi\phi}, \tag{2.44c}$$

where the Kronecker $\delta_{\psi\phi}$ vanishes except when $\psi = \phi$, in which case it evaluates to 1. A similar set of relations holds for fermionic creation and annihilation operators, with the commutators replaced with anti-commutators, such as $\{c_\psi^\dagger, c_\phi^\dagger\} \doteq c_\psi^\dagger c_\phi^\dagger + c_\phi^\dagger c_\psi^\dagger = 0$.

The actions of these operators are most naturally seen in the occupation-number or Fock basis, in which the number of particles occupying each of an orthogonal set of states is explicitly stated. For example, consider the graph P_3 of Figure 2.6(b); there are three distinct vertices on which a particle, or quantum walker, can exist. The state of M identical particles on this graph exists in the Hilbert space spanned by the states

$$(c_1^\dagger)^{m_1}(c_2^\dagger)^{m_2}(c_3^\dagger)^{m_3}|\text{vac}\rangle \doteq (c_1^\dagger)^{m_1}(c_2^\dagger)^{m_2}(c_3^\dagger)^{m_3}|0_1 0_2 0_3\rangle \doteq |m_1 m_2 m_3\rangle \tag{2.45}$$

for non-negative integers m_v subject to $\sum_v m_v = M$. For a single walker on the graph, the states $c_v^\dagger|\text{vac}\rangle$ correspond directly to the states of the vertex basis $|v\rangle$.

Chapter 3

Computing with quantum walks

The initial motivation for the study of quantum walks within the realm of quantum information lay in their analogy to classical random walks, and their algorithmic applications [47]. In particular, the fact that the quantum walk on a line spreads quadratically more quickly than a classical random walk was identified as a possible source of speed-ups in the algorithmic context of sampling and exploring structured spaces [48, 49]. This projection was realized in some of the first quantum walk applications, which were to problems with previously known quantum algorithms exhibiting speed-ups over their classical counterparts. In particular, quantum-walk search algorithms yield the quadratic speed-up exhibited by Grover's algorithm [50, 51]. In 2004 Ambainis presented a quantum-walk algorithm for the element-distinctness problem that was not only more efficient than previous non-walk-based algorithms, but in fact optimal [52]. This result was later extended to the problem of subset finding [53]. Triangle finding and its generalization to clique finding are problems related to the properties of graphs that can be solved by quantum walks on graphs [54].

Before providing a discussion of universal quantum computation in Section 3.3, two other quantum walk-based algorithms, for evaluating NAND trees and traversing glued trees, are described here in more detail to provide some intuition into the general concept of computing with quantum walks. As with triangle finding, these algorithms make use of quantum walks on graphs to answer questions about those same graphs.

3.1 Quantum walk algorithms

The first quantum walk algorithm was provided along with the introduction of the continuous-time quantum walk by Farhi and Gutmann who show that a walker traversing a tree graph can solve decision problems [37], though a quantum walker does so no more quickly than a classical one. As described in Section 1.1.2, a decision problem requires a Boolean answer (i.e. yes or no) dependent on the specified input to a particular question. For example in the Boolean satisfiability problem, or SAT, one is given a set of Boolean-OR clauses such as $\{(x_1 \text{ OR } x_2), (x_1 \text{ OR } x_2 \text{ OR NOT } x_3)\}$, and asked to determine whether there is an assignment of the variables x_i such that all clauses are satisfied simultaneously, meaning that their logical AND evaluates to TRUE. A simple example is that $(x_1 \text{ OR } x_2) \text{ AND } (x_1 \text{ OR NOT } x_2) = \text{TRUE}$ under either of the two variable assignments $(x_1, x_2) = (1, 1)$ and $(x_1, x_2) = (1, 0)$, so in this case the answer is 'yes'. The satisfiability problem with clauses of k variables is referred to as k-SAT, with 2SAT and 3SAT being of particular note because 2SAT is in P while the Cook–Levin Theorem states that 3SAT is NP-complete [55]. Indeed SAT as a whole is NP-complete since it subsumes 3SAT, and together these were the first problems shown to be so.

3.1.1 Traversing glued trees

The first quantum walk algorithm to provide a speed-up over classical methods was the traversal of randomized glued trees [56], which was an extension to the quantum walk algorithm for the traversal of glued trees [57]. The latter provides a speed-up over classical random walk algorithms, but not over all possible algorithms. A glued-trees graph is given by a pair of constant-depth trees, the leaves of which have been connected from one tree to the other. If each leaf in either tree is connected randomly to exactly two leaves of the other tree, the result is a randomized glued-trees graph. An example of each type of gluing can be seen in Figure 3.1.

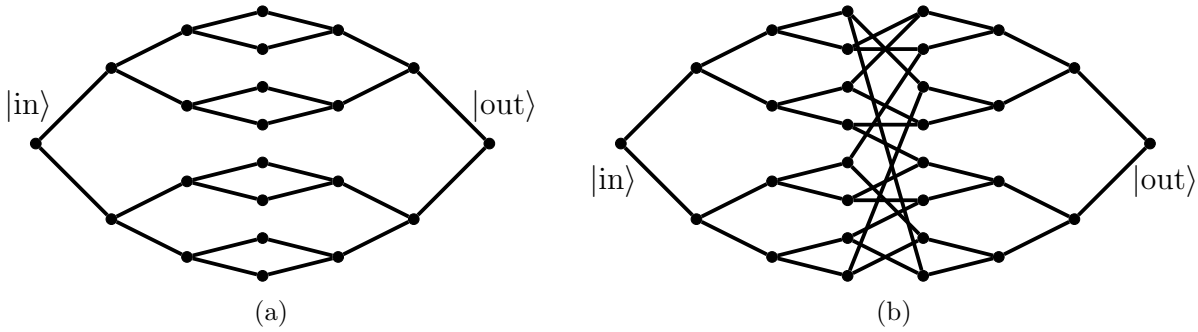


Figure 3.1: (a) Glued trees, which a quantum walk can traverse exponentially more quickly than a classical random walk. (b) A randomized gluing of the trees, which a quantum walk can traverse exponentially more quickly than any classical algorithm.

Childs, Farhi, and Gutmann introduce a problem that is solved exponentially more quickly by a quantum walk than by a classical random walk , and provide a modification to the problem such that the quantum version exponentially outperforms any possible classical algorithm . The initial problem is to traverse a pair of binary trees whose leaves have been pairwise identified, or 'glued', walking from the root of one tree to the root of the other. Figure 3.1(a) illustrates this construction with glued trees of depth $n = 3$.

Consider a classical random walker that enters the graph at the $|\text{in}\rangle$ vertex, the left-hand root of the glued trees in Figure 3.1(a). The problem is for the walker to reach the $|\text{out}\rangle$ vertex of the right-hand root in order to exit the graph. This is obviously a straightforward task if at each step the walker has complete knowledge of the graph and can always know which vertices are closer to $|\text{out}\rangle$ than its current position. To make the problem non-trivial it is instead framed in terms of a query model, wherein at each step the walker is allowed to make one query to a so-called oracle, which has complete knowledge of the graph and the position of the walker upon it, but which will tell the walker only the identities of the vertices adjacent to its current vertex. The difficulty of the problem is then proportional to the number of queries the walker must make, on average. For the first step the oracle will reveal two vertices to the walker, which then

moves one step close to the exit regardless of its choice. On each subsequent step, until it reaches the middle of the graph, the walker is given three neighbouring vertices, two of which are closer to the exit. It is therefore twice as likely to step to the right as to the left and is likely to rapidly arrive at the centre, where the leaves are glued together. At this point it is equally likely to move left or right, but regardless of the direction it travels the walker will become once again twice as likely to step toward the leaves as toward the nearest root. This tendency for the walker to prefer remaining near the leaves by a factor of 2:1 means that it is highly unlikely for the walker to reach the other end of the trees, or indeed to return to its starting point. More specifically, having begun its walk on the left-hand root of a pair of glued trees of depth n, the probability to find the walker on the right-hand root after any number of steps is less than 2^{-n}, meaning that the probability to traverse the trees in a time polynomial in n is exponentially small as a function of n.

In the quantum case, on the other hand, Childs *et al.* show that after a evolving on the glued-trees graph for a polynomial length of time after starting on the left-hand root, a continuous-time quantum walker has a probability of at least $\Omega(1/n)$ to be found on the right-hand root. While this is still not a large probability, it is polynomially rather than exponentially small in n. Therefore by executing the walk $O(n)$ times, the probability to find the right-hand root can be made arbitrarily close to unity in a polynomial time. That is, the quantum walk is exponentially faster than the classical random walk in the completion of this task. However, even within the query model a non-random classical walk can find the $|\text{out}\rangle$ vertex as quickly as the quantum walker. By remembering a list of all vertices it has already visited, a non-random classical walker can reach the central column of vertices in exactly n steps by always choosing to move to a previously unvisited vertex. It then moves into the second half of the glued trees, continuing to only transition to previously unvisited vertices. This may well bring it back to the centre, but

because the central vertices are degree 2 while the rest of the internal vertices are degree 3, this can be detected; if it happens s steps after the walker first left the middle, then by retracing the last $s/2$ steps and choosing the other path, the walker can be sure of progressing toward the exit.

This is the motivation behind the randomized gluing of trees exemplified in Figure 3.1(b). There is no classical strategy that allows $|\text{out}\rangle$ to be reached with a sub-exponential number of steps on average, because the central vertices can no longer be distinguished by their degree. Of course, employing the standard technique of evolving a quantum walker according to the unitary operator generated by the graph Hamiltonian is akin to allowing a classical walker to see its position relative to the entire graph. Comparison of such evolution with the classical query model would not be meaningful, so Childs *et al.* provide a continuous-time quantum query model under which a quantum computer equipped with a unitary oracle operator U can efficiently approximate a continuous-time quantum walk on the graph. The nature of the oracle operator is to bit-wise add the label u of the ith neighbour of vertex v to an ancillary vertex label x according to

$$U|i, v, x\rangle = |i, v, x \oplus u\rangle, \tag{3.1}$$

and with it the quantum walk simulated under the circuit model solves the glued-tree traversal problem using a number of one- and two-qubit gates and a number of calls to the oracle operation that are each polynomial in the depth n of each of the trees. More generally, it has since been shown that any algorithm presented in the continuous-time quantum query model can be simulated by a quantum circuit making discrete queries with at most a logarithmic increase in the total query time [58, 59].

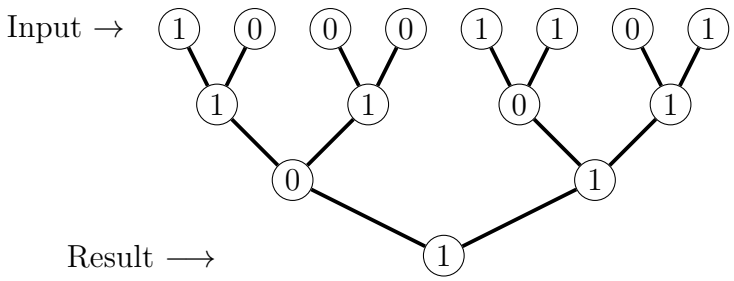


Figure 3.2: Example evaluation of a NAND tree on eight input bits. The input values are assigned to the leaves of the tree, and the vertices in each successively smaller layer contain the values equal to the logical-NAND of their neighbours in the preceding layer, until the answer appears in the root.

3.1.2 Evaluating NAND trees

While traversing a pair of glued binary trees shows with certainty that there exist tasks that a quantum walk can perform exponentially more quickly than any classical algorithm, the specific task in and of itself is not particularly interesting. A problem whose solution more obviously represents the implementation of a computation is the evaluation of NAND trees, which is an alternative formulation of 2SAT and can be stated as follows.

Given an $N = 2^n$-bit binary string, assign the bits sequentially to the leaves of an n-level binary tree. For each pair of leaves sharing a parent, assign to that parent the value given by the NAND of their values. Repeat this process, assigning to each vertex in the tree the NAND value of its two children, until the single value of the root has been determined. This value is the desired solution to the NAND-tree evaluation, an example of which can be found in Figure 3.2.

Farhi, Goldstone, and Gutmann provide a quantum-walk based algorithm for the evaluation of a NAND tree on N input bits in a time proportional to $\sqrt{N}$, whereas the optimal classical runtime is proportional to $N^{0.753} > \sqrt{N}$ [60].[1] This polynomial speed-up is small compared with the exponential speed-up of the glued-trees traversal, but it is nevertheless another example of a task that a quantum walk can accomplish in

[1]The exponent 0.753 is an approximation to the exact result of $\log_2(\frac{1+\sqrt{33}}{4})$.

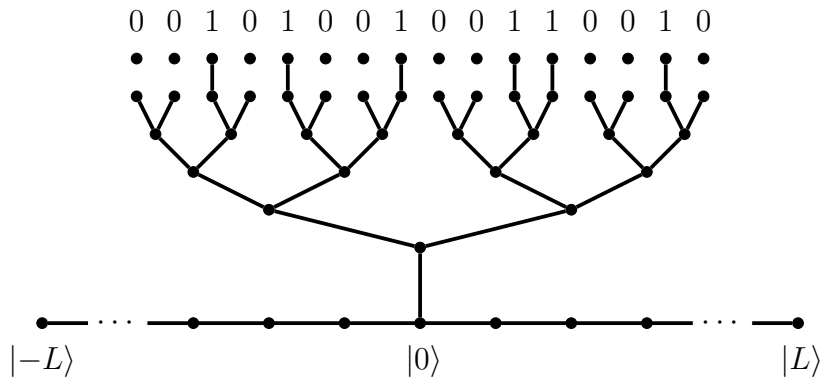


Figure 3.3: Example graph for evaluating a NAND tree with a continuous-time quantum walker.

strictly less time than any classical algorithm. Furthermore, the algorithm is of particular interest here as its method of computing foreshadows the scheme for universal quantum computation by quantum walk, described in Section 3.3.

It is straightforward to illustrate the evaluation of a NAND tree by placing values on the vertices, but there is no mechanism by which to actually ascribe values to the vertices of a simple graph. Instead, the values of the input bits are assigned by way of the connectivity of the graph. To accomplish this, an additional row of 2^n vertices is added to a binary tree, one for each leaf, and an edge is added between the leaf and associated new vertex for each input bit to be set equal to 1. Additionally, a linear graph of length $2L + 1$, for some sufficiently large L as will be discussed soon, is added to the graph with its central vertex connected to the root of the modified tree. The result is a graph similar to that in Figure 3.3, with the vertices of the lower 'rail' labelled from $-L$ to L so that the tree attaches to vertex $|0\rangle$.

Consider for a moment only the rail, which is constructed on the $2L + 1$ vertices $\{|-L\rangle, \ldots, |L\rangle\}$ with adjacency matrix

$$A_{\text{rail}} = \sum_{x=-L}^{L-1} |x\rangle\langle x+1| + \text{H.c.} \tag{3.2}$$

The Hamiltonian of the system is taken to be the negative of the adjacency matrix of the graph. In the large-L limit, the rail supports momentum eigenstates $|k\rangle$ of the form

$$\langle x|k\rangle \propto e^{\mathrm{i}kx} \tag{3.3}$$

with corresponding energies $E(k) = -2\cos(k)$, for $k \in (-\pi, \pi]$. For $k > 0$, the states $|k\rangle$ and $|-k\rangle$ correspond to plane waves of equal energy but opposite directions of propagation—right-moving and left-moving, respectively. The evaluation of the NAND tree is accomplished by the scattering of a quantum walker whose state initial state is a right-moving wave packet, incident upon the attached tree from the left-hand side of the rail. With the tree attached at its root to vertex $|0\rangle$ of the rail, the supported eigenstates of the system take the form on the rail of

$$\langle x|k\rangle = \begin{cases} \mathrm{e}^{\mathrm{i}kx} + R(k)\mathrm{e}^{-\mathrm{i}kx}, & x < 0; \\ T(k)\mathrm{e}^{\mathrm{i}kx}, & x \geq 0, \end{cases} \tag{3.4}$$

where $R(k)$ and $T(k)$ are momentum-dependent reflection and transmission coefficients for the scattering process of the walker's evolution through the NAND tree. What Farhi, Goldstone, and Gutmann show is that if $\nu \in \{0, 1\}$ is the value to which the NAND tree evaluates, then $T(\pi/2) = \nu$. That is, if the tree evaluates to $\nu = 1$ then a plane wave with momentum $k = \pi/2$ has unit transmission, and if the tree evaluates to $\nu = 0$ then the wave exhibits perfect reflection. Furthermore, eigenstates with momenta near $k\pi/2$ also have transmission coefficients close to ν. Therefore, remarkably, the result of the evaluation can be determined simply by looking for a quantum walker on the right-hand side of the rail. A quantum walker initialized as a propagating wave packet initially on the left-hand rail with a momentum profile tightly peaked near $\pi/2$ has a high probability to be found at a later time in a similarly shaped packet on either the right-hand or left-hand

rail, depending on whether the NAND tree evaluates to 1 or 0.

Based on the group velocity of the wave packet,

$$v(k) = \frac{\mathrm{d}E}{\mathrm{d}k} = 2\sin(k), \tag{3.5}$$

the quantum walker traverses the rail with speed $v(\pi/2) = 2$ and so travels a distance L—from the centre of the left-hand rail to either the centre of the right-hand side or back to its starting point—in a time $L/2$. It turns out that the probability of successfully evaluating the NAND tree in this way can be made arbitrarily close to 1 when L, and therefore a run-time, is proportional to $\sqrt{N}$. This ensures that the wave packet consists primarily of eigenstates with energies sufficiently close to $\pi/2$ that their transmission coefficients are within the desired error tolerance of ν. Thus not only does this method of scattering a quantum walker off of a graph with topology dictated by a problem yield a solution to that problem, but it does so more quickly than the best possible classical algorithm. As in the glued-trees example, and indeed all quantum walk algorithms, this result can again be simulated under the quantum query model in a similar time.

3.2 Graph scattering

Finding graphs that exhibit reflection and transmission coefficients that transform a quantum walker such that its final state encodes the result of a desired computation is central to the continuous-time quantum-walk model of universal computation. Motivated by this fact, Varbanov and Brun provide a formalism for the analysis of scattering off of arbitrary finite graphs with arbitrary numbers of attached tails [61], the relevant features of which are summarized here. Let G be a finite graph with adjacency matrix A_G, on N vertices $\{v\}_{v=1}^{N}$, each associated to a normalized state $|v\rangle$ in the Hilbert space of the graph, $\mathscr{H}_G$. These states are pairwise orthogonal, obeying $\langle u|v\rangle = \delta_{uv}$, and thus provide

a basis for the N-dimensional $\mathscr{H}_G$. To each vertex v are attached $m_v \geq 0$ tails, which are semi-infinite path graphs on additional vertices $\{x_{v,m}\}_{x=1}^{\infty}$. The index v specifies the vertex to which the tail is attached, and $m \in \{1, \ldots, m_v\}$ indexes the tail among those attached to vertex v. That is, $|x_{v,m}\rangle$ is the basis state associated with vertex x of the mth tail attached to vertex v. Each tail is described by an adjacency matrix

$$A_{\text{tail}}^{(v,m)} \doteq \sum_{x=1}^{\infty} |(x+1)_{v,m}\rangle\langle x_{v,m}| + \text{H.c.} \tag{3.6}$$

and the full scattering system, comprising the scattering graph G plus tails, has adjacency matrix

$$A \doteq A_G + \sum_{v=1}^{N} \sum_{m=1}^{m_v} \left(A_{\text{tail}}^{(v,m)} + |v\rangle\langle 1_{v,m}| + |1_{v,m}\rangle\langle v| \right), \tag{3.7}$$

where the sum over m contributes no terms whenever $m_v = 0$. With the scattering graph defined, the Hamiltonian governing the evolution of a quantum walk on it is simply $\mathcal{H} = -A$. The goal now is to characterize the eigenstates of this operator, in particular their reflection and transmission coefficients with respect to G when there is an incoming (i.e. toward G) momentum component on exactly one tail.

Denote by $|k_{v,m}\rangle$ such an energy eigenstate with momentum $k \in (0, \pi)$ incoming on tail m of vertex v. On this tail, the state takes the form

$$\langle x_{v,m} | k_{v,m} \rangle = \mathrm{e}^{-\mathrm{i}kx} + R_{v,m}(k)\mathrm{e}^{\mathrm{i}kx}, \tag{3.8}$$

and on the other tails, specified by ordered pairs $(u, l) \neq (v, m)$, there is at most an outgoing component,

$$\langle x_{u,l} | k_{v,m} \rangle = T_{u,l}^{(v,m)}(k) e^{\mathrm{i}kx}. \tag{3.9}$$

Note the apparent sign change in the exponentials of Equation (3.8) as compared to those appearing in the NAND-tree case in Equation (3.4). This is due to the fact that

the incoming component on the NAND-tree has support on vertices labelled by negative positions while here all tail vertices are labelled by positive numbers, which increase away from the scattering graph.

Finally, the restriction of the momentum state to the graph G is denoted $|k_{v,m}\rangle^G$ so that

$$|k_{v,m}\rangle = |k_{v,m}\rangle^G + \sum_{x=1}^{\infty} \left[\mathrm{e}^{-\mathrm{i}kx} + R_{v,m}(k)\mathrm{e}^{\mathrm{i}kx}\right] |x_{v,m}\rangle + \sum_{(u,l)\neq(v,m)} \sum_{x=1}^{\infty} T_{u,l}^{(v,m)}(k) e^{\mathrm{i}kx} |x_{u,l}\rangle. \tag{3.10}$$

The behaviour of the state away from the scattering graph on any of the tails dictates that the corresponding energy must be $E(k) = -2\cos(k)$. With these definitions in place, Varbanov and Brun show that the restriction of the eigenstate to G satisfies the equation

$$\left(\mathrm{e}^{\mathrm{i}k} \sum_{u=1}^{N} m_u |u\rangle\langle u| - 2\cos(k) I_N - \mathcal{H}_G\right) |k_{v,m}\rangle^G = 2\mathrm{i}\sin(k)|v\rangle, \tag{3.11}$$

where I_N is the N-dimensional identity matrix. Once this set of N coupled equations has been solved for $|k_{v,m}\rangle^G$, the scattering coefficients can be read off as

$$R_{v,m}(k) = \langle v\,|k_{v,m}\rangle^G - 1, \tag{3.12a}$$

$$T_{u,l}^{(v,m)}(k) = \langle u\,|k_{v,m}\rangle^G. \tag{3.12b}$$

A further consideration when analysing the scattering of a quantum walker by a graph, and that is the *effective length* of the scattering graph. Clearly as a walker is scattered by a graph G it must take some finite time to travel through it, which is captured by the effective length of the graph from the incoming tail to each of the outgoing ones.

The simplest case is that of two tails, attached to the two ends of a linear graph with

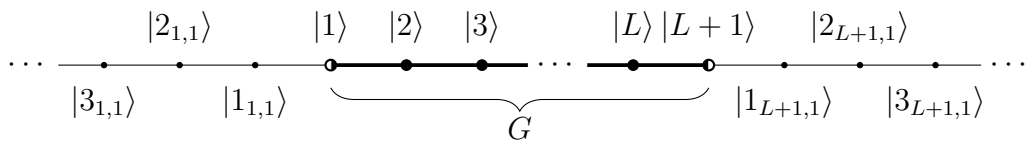


Figure 3.4: Demonstration of the effective length of a graph. A linear graph G of L segments, with a semi-infinite tail attached at each end, should have an effective length $\ell_G(k)$ equal to the actual length L of the graph in this simple case.

L segments. That is, G has $L+1$ vertices with adjacency matrix

$$A_G = \sum_{v=1}^{L} |v+1\rangle\langle v| + \text{H.c.} \tag{3.13}$$

and the two semi-infinite tails are attached at $|1\rangle$ and $|L+1\rangle$. This setup is shown in Figure 3.4. The definition of the scattering system fixes the phase of a plane wave on the first vertex of G so that it vanishes, since $\langle 1\,|k_{v,m}\rangle = 1$, and on the last vertex $\langle L+1\,|k_{v,m}\rangle = T^{(1,1)}_{L+1,1}(k)$. However since this system is simply an infinite line, and given the phase on vertex $|1\rangle$, it must also be the case that $\langle L+1\,|k_{v,m}\rangle = \mathrm{e}^{\mathrm{i}kL}$. That this phase shift over a length L is equal to the phase of the transmission coefficient through the same length motivates the definition of the effective length of a graph between incoming tail (v,m) and outgoing tail (u,l) more generally as [62]

$$\ell^{(v,m)}_{u,l}(k) \doteq \frac{\mathrm{d}}{\mathrm{d}k} \arg T^{(v,m)}_{u,l}(k). \tag{3.14}$$

Note that the effective length need not be the same between arbitrary pairs of attachment vertices.

3.3 Universal quantum walk-based computation

To show that quantum walks are universal for quantum computation, Childs proposed a model influenced by the scattering method of calculation discussed in the previous sections [62]. A sufficiently long rail—infinite, in the ideal case—is assigned to each computational basis state to be simulated, so that for an n-qubit calculation, 2^n rails are required. A quantum walker on the graph is then able to encode an n-qubit state in its position, with one rail assigned to each of the computational basis states. As with the NAND-tree evaluation, a quantum walker is initialized on the left-hand 'input' side of the rails, and then propagates to the right for a set amount of time, at which point a measurement on the right-hand 'output' side of the rails yields the result of the computation. Also akin to the NAND-tree algorithm, it is the scattering properties of graphs attached to the rails between the input and output sections that enable the final state of the walker to encode the result. These graphs, termed *widgets*, are simple, finite, and small, with the largest of the set proposed by Childs having only seven vertices.

3.3.1 Simulating a single-qubit gate

The general setup for the simulation of a single qubit is portrayed in Figure 3.5. Given a finite graph G, four semi-infinite tails are attached to it. The graph simulates a single-qubit gate if two of them can be identified as the computational $|\underline{0}\rangle$ rail, and the other two as the $|\underline{1}\rangle$ rail. That is, if the position of the walker is measured and it is found to be on a vertex of the $|\underline{0}\rangle$ rail then the computational state of the single encoded qubit is $|\underline{0}\rangle$, and likewise for the $|\underline{1}\rangle$ rail. If the walker is located on a vertex of the widget graph G, then in general no computational state is encoded. This ability of the system to evolve through states that do not encode quantum information, yet return to the computational space and continue the implementation of an algorithm is in contrast with the standard circuit model, in which at every instant of evolution the qubits encode a computational

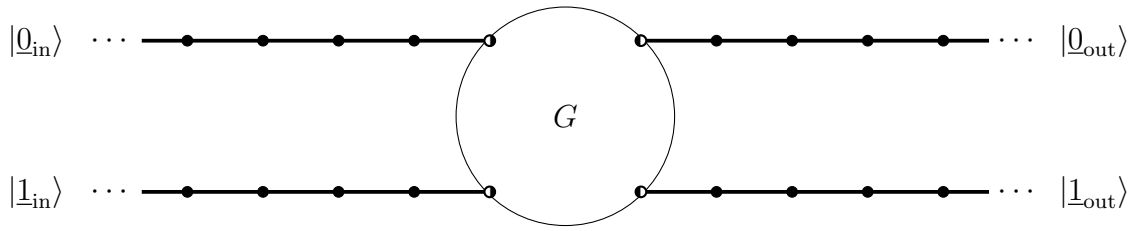


Figure 3.5: Schematic representation of the setup to implement a single-qubit gate by graph scattering. G takes the place of a finite graph to which four tails have been attached; half-open circles represent input and output vertices, when open on the left and right, respectively. Note that the tails need not be attached to four distinct vertices.

state. This difference arises again in Chapter 6, where it is highlighted in more detail.

Consider a quantum walker in a wave-packet state localized on the $|\underline{0}_{\text{in}}\rangle$ tail and propagating toward the graph G, with a momentum profile tightly peaked about some momentum k. In general the walker will scatter into a superposition of wave packets propagating away from G along all four tails. Letting the indices i and j run over the tail labels $\{\underline{0}_{\text{in}}, \underline{0}_{\text{out}}, \underline{1}_{\text{in}}, \underline{1}_{\text{out}}\}$, this transition of probability amplitudes from an incoming wave packet on one tail to a superposition of outgoing packets on four can be described by the scattering matrix

$$S(k) \doteq \left(\sum_i R_i(k)|i\rangle\langle i| + \sum_{\substack{i,j \\ i\neq j}} T_i^{(j)}(k)|i\rangle\langle j| \right), \tag{3.15}$$

which is unitary [61]. For specific combinations of G and k however, there will be no reflection back along $|\underline{0}_{\text{in}}\rangle$ and no transmission to the $|\underline{1}_{\text{in}}\rangle$ tail. In such cases, there is perfect transfer of the walker from $|\underline{0}_{\text{in}}\rangle$ to a normalized superposition of $|\underline{0}_{\text{out}}\rangle$ and $|\underline{1}_{\text{out}}\rangle$. If for the same combination of G and k a similar walker propagating toward G along the $|\underline{1}_{\text{in}}\rangle$ rail also transfers perfectly to the two output rails, then the unitary evolution of the walker through the graph can also be interpreted as a transformation on the qubit

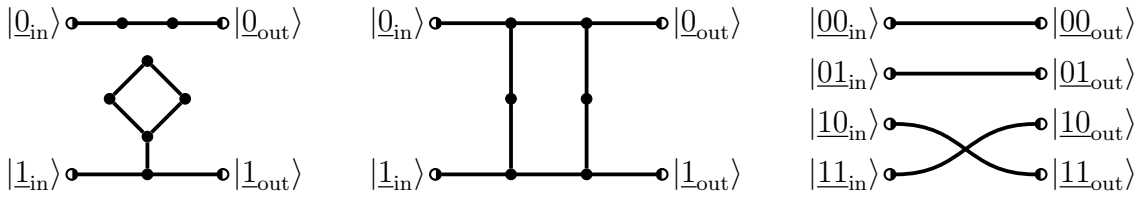
encoded by the position of the walker, under the unitary operator

$$U = \begin{pmatrix} T_{\underline{0}_{\text{out}}}^{(\underline{0}_{\text{in}})} & T_{\underline{0}_{\text{out}}}^{(\underline{1}_{\text{in}})} \\ T_{\underline{1}_{\text{out}}}^{(\underline{0}_{\text{in}})} & T_{\underline{1}_{\text{out}}}^{(\underline{1}_{\text{in}})} \end{pmatrix}. \tag{3.16}$$

There is one more ingredient to this scheme, that of the effective lengths of the paths through the graph. As described in Section 3.2, the effective length between two of the four tails attached to G captures the fact that a walker takes a finite amount of time to propagate through the graph from one to the other, if indeed it does so at all, and is a momentum-dependent quantity. In order to maintain the spatial coherence of the walker as it propagates through G from the input to the output tails, one of two situations must occur. Either G transfers probability from each input to a superposition of the outputs, in which case the four effective lengths through the graph must all be equal, or it transfers probability from each input to a single output, in which case the two lengths must be equal. Note that while probability cannot transfer between two vertices if there is no path between them, the existence of a path does not guarantee that probability will transfer at all momenta.

3.3.2 A universal single-qubit gate set

This section briefly summarizes the widgets used in Reference [62] to create a set of gates universal for single-qubit computation. There are two sets of widgets employed by the scheme, serving distinct purposes. The first, shown in Figure 3.6, contains the computational widgets that provide a set of gates universal for quantum computation in the ideal case of infinitely long tails supporting plane-wave states. The second, presented in Figure 3.7, is required when the tails are of finite length and the state of the walker is a propagating wave packet rather than a plane wave, or even if it is initialized on a single vertex in a superposition of all momentum states.



(a) Phase shifter (b) Basis changer (c) CNOT

Figure 3.6: Computational widgets providing a universal gate set, due to Childs [62]. The phase shifter and basis changer together are universal for single-qubit operations, and the addition of the CNOT gate results in a set universal for quantum computation.

(a) Momentum filter (b) Momentum separator

Figure 3.7: Non-computational widgets of Reference [62]. (a) The momentum filter has perfect transmission from $|\text{in}\rangle$ to $|\text{out}\rangle$ only at momenta $\pi/4$ and $3\pi/4$; other momenta are either reflected or transmitted to the semi-infinite tail extending upward, and chaining m such filters in series results in a transmission probability exponentially small in m for all momenta not equal to $\pi/4$ or $3\pi/4$. (b) The momentum separator also has perfect transmission from $|\text{in}\rangle$ to $|\text{out}\rangle$ at both $k = \pi/4$ and $k = 3\pi/4$, but crucially the effective length seen by $k = 3\pi/4$ is an order of magnitude larger than that seen by $k = \pi/4$, so this widget introduces a spatial separation between the two momentum components not removed by the momentum filter.

The computational widgets in Figure 3.6 all have perfect transmission at momentum $k = \pi/4$. In his original formulation, Childs takes the system Hamiltonian to be equal to the adjacency matrix, $\mathcal{H} = A$, rather than its negative; this results in his discussion's invoking $k = -\pi/4$, but being functionally equivalent to the current summary. The phase shifting widget of Figure 3.6(a) has an effective length of $\ell(\pi/4) = 3$ from each input to its corresponding output, but with transmission coefficients

$$T_{\underline{0}_{\text{out}}}^{(\underline{0}_{\text{in}})}(\pi/4) = \mathrm{e}^{3\mathrm{i}\pi/4}, \tag{3.17a}$$

$$T_{\underline{1}_{\text{out}}}^{(\underline{1}_{\text{in}})}(\pi/4) = \mathrm{e}^{\mathrm{i}\pi/2}, \tag{3.17b}$$

the portion of the walker on the $|\underline{1}\rangle$ rail acquires a phase of $-\pi/4$ relative to that on the $|\underline{0}\rangle$ rail. Here $T_{\underline{b}}^{(\underline{a})}(k)$ is the transmission coefficient for a plane wave of momentum k scattering from tail $\underline{a}$ to tail $\underline{b}$ [cf. Equation (3.9)]. The encoded computational state of a walker incident on this widget in initial state $|\psi_{\text{in}}\rangle = \alpha|\underline{0}_{\text{in}}\rangle + \beta|\underline{1}_{\text{in}}\rangle$ is transformed by the computational unitary operator

$$U_{\text{PS}} = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-\mathrm{i}\pi/4} \end{pmatrix} \tag{3.18}$$

to the state $|\psi_{\text{out}}\rangle = \alpha|\underline{0}_{\text{out}}\rangle + \mathrm{e}^{-\mathrm{i}\pi/4}\beta|\underline{1}_{\text{out}}\rangle$. The phase gate U_{PS} differs in the sign of the applied phase from that of Reference [62], since therein the transmission coefficients of Equations (3.17a) and (3.17b) are $\mathrm{e}^{-3\mathrm{i}\pi/4}$ and $\mathrm{e}^{-\mathrm{i}\pi/2}$ respectively, each differing in the sign of its phase due again to Childs' choice of Hamiltonian as $\mathcal{H} = A$. This difference is unimportant to the computational scheme.

The other single-qubit widget of the scheme, seen in Figure 3.6(b), simulates a basis-

changing gate. Its transmission coefficients from the input to output vertices are

$$\left.\begin{pmatrix} T_{\underline{0}_{\text{out}}}^{(\underline{0}_{\text{in}})} & T_{\underline{0}_{\text{out}}}^{(\underline{1}_{\text{in}})} \\ T_{\underline{1}_{\text{out}}}^{(\underline{0}_{\text{in}})} & T_{\underline{1}_{\text{out}}}^{(\underline{1}_{\text{in}})} \end{pmatrix}\right|_{k=\pi/4} = -\frac{1}{\sqrt{2}} \begin{pmatrix} \mathrm{e}^{\mathrm{i}\pi} & \mathrm{e}^{3\mathrm{i}\pi/2} \\ \mathrm{e}^{3\mathrm{i}\pi/2} & \mathrm{e}^{\mathrm{i}\pi} \end{pmatrix} \tag{3.19}$$

which, up to an overall phase factor, defines the computational unitary effected by the basis-changing widget,

$$U_{\text{BC}} \doteq -\frac{1}{\sqrt{2}} \begin{pmatrix} \mathrm{i} & 1 \\ 1 & \mathrm{i} \end{pmatrix}. \tag{3.20}$$

Together the single-qubit gates U_{PS} and U_{BC} form a single-qubit instruction set, capable of efficiently simulating any single-qubit gate since $\mathrm{i}U_{\text{PS}}^6 U_{\text{BC}} U_{\text{PS}}^6 = H$, and the Hadamard operator together with the $\pi/8$ phase gate U_{PS}^7 generate SU(2) [4].

3.3.3 Simulating an entangling gate

Perhaps surprisingly, introducing entanglement between qubits simulated by a quantum walk can be more straightforward than implementing a single-qubit gate. This is due to the fact that a single quantum walker encodes computational states in its position, and therefore each computational state can be addressed individually; adding a phase to those rails encoding states in which the value of a pair of qubits is $|\underline{11}\rangle$ simulates a CPHASE gate between them, for example. Even more trivial to implement is the CNOT gate, which forms part of the universal gate set in Reference [62].

More explicitly, consider the simulation of two qubits. The walker propagates on four rails, one for each of the computational basis states $|\underline{00}\rangle$, $|\underline{01}\rangle$, $|\underline{10}\rangle$, and $|\underline{11}\rangle$, and simply interchanging the $\underline{10}$ and $\underline{11}$ rails—or merely keeping track of a relabelling of them—will effect a CNOT gate with the first qubit as control and the second as target. This is depicted in Figure 3.6(c), along with the widgets constituting a universal set of

gates which together show that continuous-time quantum walks are indeed universal for quantum computation.

3.3.4 Computing on finite graphs

The general formulation of the continuous-time quantum-walk model of computation described so far depends on semi-infinite tails in order to support plane-wave momentum eigenstates. While this simplifies the analysis, a scheme that required infinitely many vertices to implement, even in theory, would not be of practical interest. However, Childs shows that the tails attached to the computational graph can be truncated at a length polynomial in m, the number of gates to be simulated, in exchange for an increase in the expected runtime of the simulation that is also polynomial in m. It is intuitive that such a truncation should be possible with minimal effect on the dynamics of the walker, because the magnitude of the group velocity in Equation (3.5) is bounded above by $v(\pi/2) = 2$; boundary effects will take a finite time to reach a walker initialized sufficiently far from the truncation point. Furthermore, it is possible to initialize the walker on a single vertex to avoid the requirement of constructing a propagating wave packet. These two tasks, tail truncation and single-vertex initialization, are collectively made possible without compromising the computational power of the walker by the momentum filter and separator widgets shown in Figure 3.7. The filter widget transmits perfectly only at momenta $k = \pi/4$ and $k = 3\pi/4$, while the separator introduces a delay between these two components. Thus a time can be determined at which the probability amplitude of the $\pi/4$ momentum component will be spatially isolated on the output tails.

More specifically, the momentum filter of Figure 3.7(a) has perfect transmission between its $|\text{in}\rangle$ and $|\text{out}\rangle$ vertices at momenta $k = \pi/4$ and $k = 3\pi/4$, while other momentum components are either at least partially reflected, or transmitted along the semi-infinite tail extending upward from the widget and away from the computational graph.

When the widget is repeated m_F times in sequence, the magnitude of the transmission coefficient from the first $|\text{in}\rangle$ to the last $|\text{out}\rangle$ is exponentially small in m_F for momenta not close to $\pi/4$ or $3\pi/4$ [62]. The computational widgets also have perfect transmission at both $k = \pi/4$ and $k = 3\pi/4$, however their behaviours are different at these to momentum values. For example, the phase-shifting widget predictably shifts the phase of a walker on the $|\underline{1}\rangle$ rail by $-3\pi/4$ relative to the $|\underline{0}\rangle$ rail, as opposed to by $-\pi/4$. In general this might not present a problem since different momentum components propagate with different velocities and thus become spatially separated, however the two momenta in question happen to have the same group velocity, $v(\pi/4) = v(3\pi/4) = \sqrt{2}$.

The momentum separator widget in Figure 3.7(b) is thus introduced to spatially separate the desirable momentum components near $k = \pi/4$ that are simulating the computation from the undesirable components near $k = 3\pi/4$, which are not. This widget has perfect transmission from $|\text{in}\rangle$ to $|\text{out}\rangle$ at both $k = \pi/4$ and $k = 3\pi/4$ (as well as at $k = \pi/2$) but crucially the effective length of the widget differs at the two momenta of interest: $\ell(\pi/4) = 2(7 - 4\sqrt{2}) \approx 2.686$, while $\ell(3\pi/4) = 2(7 + 4\sqrt{2}) \approx 25.31$. Thus with this widget the momentum components participating in the quantum computation can be spatially isolated from the other transmitted components.

An n-qubit quantum circuit containing m gates from the universal set $\{U_{\text{PS}}, U_{\text{BC}}, \text{CNOT}\}$ can be simulated by a continuous-time quantum walker on a graph constructed as follows. The computational graph comprises the widgets corresponding to the gates of the circuit, to which are attached 2^n input and 2^n output tails, labelled by the computational basis states. Between the computational graph and the $|\underline{0}\rangle^{\otimes n}$ input tail, $m_F \doteq \log(\Theta(m^2))$ momentum-filter widgets are inserted, followed by a momentum separator. A measurement can be timed such that if it finds the walker on the output rails then it must have arrived there with momentum $\pi/4$, and the computation succeeds. If the walker is located elsewhere in the graph, the system must be re-initialized and the computation

run again. The probability of success for a single run is at least $\Omega(1/m^2)$ [62], so the computation is expected to complete successfully in a number of runs—and therefore a time—polynomial in the number of gates m.

3.4 Computing with discrete-time quantum walks

Since a discrete-time quantum walk proceeds as a sequence of unitary operations without reference to any underlying generating Hamiltonian, the eigenstate analysis of graph scattering processes used to describe the continuous-time walk does not transfer directly to analysis of the discrete case. Nevertheless, the rails–as–computational–states model provides a good starting point for seeking a computationally universal discrete-time quantum walk. Indeed, this is the route taken by Lovett *et al.* in the first demonstration of a universal quantum computation scheme using discrete-time quantum walks [63].

The primary challenge to the creators of this scheme was the fact that while it is trivial to create a discrete-time quantum walk that propagates in only one direction on a line—simply use the Pauli-X operator as the coin and initialize the walker on one vertex with its coin in a basis state of X—its direction of propagation cannot be so well controlled on a graph containing vertices of degree greater than two. Since the position states associated to vertices of the graph are not conjugate to momentum states on the graph, one cannot simply construct a propagating wave packet for the initial state of the walker. To surmount this problem, Lovett *et al.* began by investigating scenarios under which a discrete-time walker can undergo perfect state transfer (cf. Section 2.4) on graphs with degree greater than two.

Their discrete-time analogue to the degree-two wires of Figure 3.5 employs graphs with multiple edges. A line of vertices with two edges between each, as shown in Figure 3.8, with the four-dimensional Grover coin

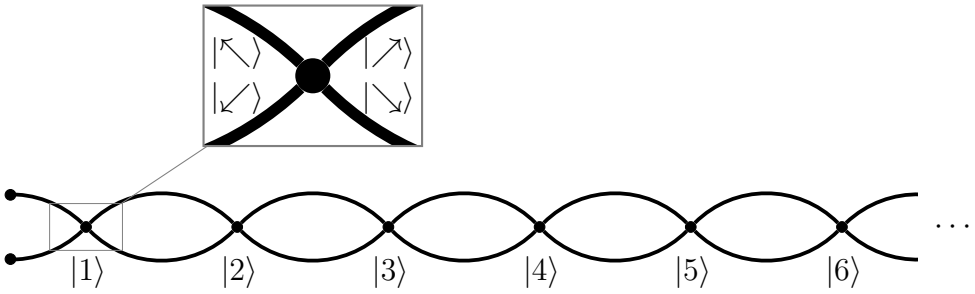


Figure 3.8: Quantum wire for a discrete-time quantum walker. A walker initialized on vertex $|1\rangle$ can be made to propagate to the right, remaining localized on a single vertex after each step. (Inset) Labelling scheme for the four states of the coin on each vertex of degree four.

$$C_{\phi}^{(4)} \doteq \frac{\mathrm{e}^{\mathrm{i}\phi}}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \tag{3.21}$$

acting at each vertex, will propagate a walker in an appropriately chosen initial state in one direction along the quantum wire with certainty. Specifically, in the vertex basis $\{|\nwarrow\rangle, |\swarrow\rangle, |\nearrow\rangle, |\searrow\rangle\}$ defined graphically in Figure 3.8 in terms of the direction the four edges leave each vertex, one step of the discrete-time quantum walk is given by

$$SC_{\phi}^{(4)}\left[|x\rangle \otimes \frac{1}{\sqrt{2}}\left(|\nwarrow\rangle + |\swarrow\rangle\right)\right] = |x+1\rangle \otimes \frac{\mathrm{e}^{\mathrm{i}\phi}}{\sqrt{2}}\left(|\nwarrow\rangle + |\swarrow\rangle\right) \tag{3.22}$$

and repeated application of the walk operator $SC^{(4)}$ transfers the walker perfectly one vertex to the right, acquiring a phase ϕ each time.

With a scheme in place for the propagation of a walker along a rail, it remains only to find a set of gates that simulates universal quantum computation. As with the continuous-time proposal, the generation of computational entanglement comes essentially for free in the form of the CNOT gate that results from interchanging the $|\underline{10}\rangle$ and $|\underline{11}\rangle$ rails, as shown in Figure 3.9. Along with this, Lovett *et al.* provide, coincidentally, the same

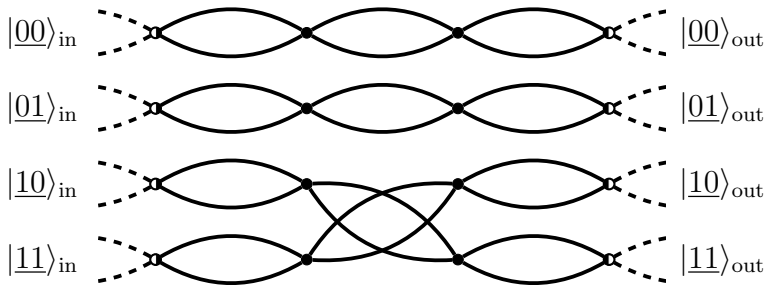


Figure 3.9: CNOT widget of the discrete-time quantum walk model. Interchanging the quantum wires carrying two of the basis states results in the implementation of an X gate on the second qubit only if the first qubit is in the state $|\underline{1}\rangle$.

universal single-qubit gate set as Childs does, though with different graph topologies and of course a different evolution scheme.

The discrete-time phase gate, presented in Figure 3.10, adds vertices of degree two to the graph, and uses the Pauli-X gate for the coin operator on them:

$$C^{(2)} \doteq X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{3.23}$$

Each rail transfers a walker from $|l_i\rangle$ to $|r_i\rangle$ in two steps, but while the walker on the upper rail picks up two $\mathrm{e}^{\mathrm{i}\phi}$ phase factors from the coin $C^{(4)}$, the walker on the lower rail acquires only the one as it leaves $|l_1\rangle$ since $C^{(2)}$ does not depend on ϕ. Thus a walk initially in the superposition

$$|\psi_{\text{in}}\rangle = (\alpha|l_0\rangle + \beta|l_1\rangle) \otimes \frac{1}{\sqrt{2}} (|\nwarrow\rangle + |\swarrow\rangle), \tag{3.24}$$

with $|\alpha|^2 + |\beta|^2 = 1$, propagates through the phase gate in two steps of the walk, ending up in the state

$$|\psi_{\text{out}}\rangle = \left(\alpha|r_0\rangle + \beta\mathrm{e}^{-\mathrm{i}\phi}|r_1\rangle\right) \otimes \frac{1}{\sqrt{2}} (|\nwarrow\rangle + |\swarrow\rangle), \tag{3.25}$$

up to an unimportant overall phase. That is, the widget in Figure 3.10 takes two time

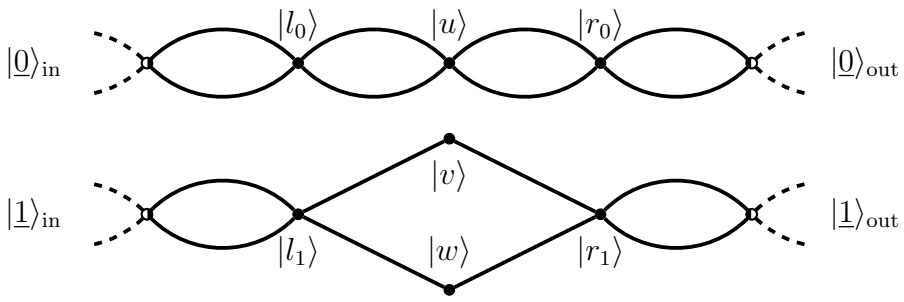


Figure 3.10: Phase widget of the discrete-time quantum walk model. A walker incident from the left traverses the two rails in the same number of steps, but acquires a different phase when it passes $|u\rangle$ as compared to when it crosses $|v\rangle$ and $|w\rangle$.

steps to effect the computational unitary

$$U_{\text{PS}} = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-\mathrm{i}\phi} \end{pmatrix}. \tag{3.26}$$

For their universal gate set Lovett *et al.* specify $\phi = -\pi/4$.

The final requirement is a basis-changing gate to turn each single-qubit computational basis state into a superposition of the two. This is achieved with a vertex of degree eight at which all of the input and output wires for a single qubit meet, as shown in Figure 3.11. The structure itself is straightforward but as with finding a mechanism by which to propagate the walker in a single direction, the key to the implementation of the basis-changing gate lies in the choice of the coin, defined in Reference [63] to be

$$C^{(8)} \doteq \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & \mathrm{i} & \mathrm{i} & -1 \\ 0 & 0 & 0 & 0 & \mathrm{i} & 1 & -1 & \mathrm{i} \\ 0 & 0 & 0 & 0 & \mathrm{i} & -1 & 1 & \mathrm{i} \\ 0 & 0 & 0 & 0 & -1 & \mathrm{i} & \mathrm{i} & 1 \\ \mathrm{i} & -1 & 1 & \mathrm{i} & 0 & 0 & 0 & 0 \\ -1 & \mathrm{i} & \mathrm{i} & 1 & 0 & 0 & 0 & 0 \\ 1 & \mathrm{i} & \mathrm{i} & -1 & 0 & 0 & 0 & 0 \\ \mathrm{i} & 1 & -1 & \mathrm{i} & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.27}$$

The basis in which this operator is expressed is the canonically numbered unit vectors

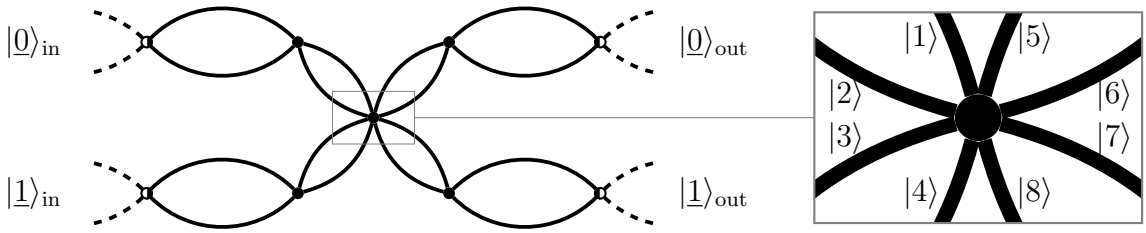


Figure 3.11: Basis-changing widget of the discrete-time quantum walk model. Mixing between the two computational rails is achieved with a vertex of degree $d = 8$, at which the coin $C^{(8)}$ defined in the text puts a walker incoming along the left-hand dual edges attached to either computational rail into a superposition of all four right-hand edges representing both computational rails. The subsequent shift operator propagates the walker one step to the right, now in a different superposition of the two computational rails.

$|1\rangle \doteq (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)^T$, $|2\rangle \doteq (0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)^T$, and so on, with the states assigned to the eight-dimensional coin degree of freedom as depicted in the inset of Figure 3.11. An incoming walker encoding the computational state $\alpha|\underline{0}\rangle + \beta|\underline{1}\rangle$ arrives on the degree-eight vertex $|v^{(8)}\rangle$ in the state

$$|\psi_{\text{in}}\rangle = |v^{(8)}\rangle \otimes \frac{1}{2}\left[\alpha(|1\rangle + |2\rangle) + \beta(|3\rangle + |4\rangle)\right], \tag{3.28}$$

which the coin $C^{(8)}$ transforms into

$$|\psi_{\text{out}}\rangle = |v^{(8)}\rangle \otimes \frac{\mathrm{e}^{3\mathrm{i}\pi/4}}{\sqrt{2}}\left[(\alpha - \mathrm{i}\beta)(|5\rangle + |6\rangle) - i(\alpha + \mathrm{i}\beta)(|7\rangle + |8\rangle)\right] \tag{3.29}$$

The subsequent shift operation propagates the walker another step to the right, back onto the computational rails in a state encoding $(\alpha - \mathrm{i}\beta)|\underline{0}\rangle - \mathrm{i}(\alpha + \mathrm{i}\beta)|\underline{1}\rangle$, up to an overall phase. That is, the walker undergoes the computational basis-changing operation

$$U_{\text{BC}} \doteq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -\mathrm{i} \\ -\mathrm{i} & 1 \end{pmatrix}, \tag{3.30}$$

the same gate as obtained in the continuous-time scheme, Equation (3.20), up to an overall phase factor. In combination with the phase shifting widget, this basis changer forms an instruction set for single-qubit computation by discrete-time quantum walk.

Chapter 4

Single-qubit gates by graph scattering

Childs' proof of the quantum-computational universality of quantum walks introduces the concept of treating a finite graph as a scattering centre when semi-infinite tails are attached to it [62]. Motivated by this result, Varbanov and Brun derive a set of equations with which the reflection and transmission coefficients can be computed given an arbitrary finite simple graph, with an arbitrary number of tails attached to any or all of its vertices [61]. The combination of these results—a model for computing by scattering walkers off of graphs, and a prescription for calculating the behaviour of such a walker incident on any finite simple graph—raises some obvious questions. For example, do other universal gate sets exist at momentum $k = \pi/4$? What about at other momenta? Does increasing the number of vertices in the scattering-centre graph increase the number of graphs that simulate quantum gates, or does the generally greater complexity of the internal structure of the graphs restrict many of them from exhibiting perfect transmission from the input rails to the output ones?

Motivated by questions such as these, I collaborated with Benjamin Blumer to exhaustively search the set of non-isomorphic simple graphs on nine or fewer vertices over a range of momenta for those resulting in scattering coefficients that could be interpreted as implementing a single-qubit gate on a quantum walker encoding a computational state. The search was accomplished numerically with a computer code written by Mr. Blumer, whom I advised during the process. I then added routines to filter out those results that appeared to provide computational unitaries but contained differing effective lengths for the paths through the graph, as described in greater detail in Section 4.1.2. My contribution also consisted in the analysis of the data output by the code to determine from a set

of scattering coefficients a meaningful description of the associated single-qubit gate, further aggregation of the elements of data set as detailed in Section 4.1.3, and the writing of the manuscript that became Reference [64] and the basis for the current chapter.

4.1 Searching for computational unitaries

To treat a graph-scattering process as a single-qubit gate—that is, as a unitary operation on the encoded computational space—only graphs with exactly four tails attached to them are considered, and the tails $\{(v_i, m_i)\}_{i=1}^{4}$ are relabelled as $\{\underline{b}_{\text{in}}, \underline{b}_{\text{out}}\}_{b=0}^{1}$. It is also notationally convenient to identify momentum states on the tails as, for example, $|k, \underline{0}_{\text{in}}\rangle$ as opposed to $|k_{\underline{0}_{\text{in}}}\rangle$. In general a walker incoming on tail $\underline{0}_{\text{in}}$ scatters according to

$$|k, \underline{0}_{\text{in}}\rangle \mapsto R_{\underline{0},\text{in}}(k)|k, \underline{0}_{\text{in}}\rangle + T_{\underline{1},\text{in}}^{(\underline{0},\text{in})}(k)|k, \underline{1}_{\text{in}}\rangle + \sum_{b=0}^{1} T_{\underline{b},\text{out}}^{(\underline{0},\text{in})}(k)|k, \underline{b}_{\text{in}}\rangle, \tag{4.1a}$$

and likewise $|k, \underline{1}_{\text{in}}\rangle$ is transformed as

$$|k, \underline{1}_{\text{in}}\rangle \mapsto R_{\underline{1},\text{in}}(k)|k, \underline{1}_{\text{in}}\rangle + T_{\underline{0},\text{in}}^{(\underline{1}_{\text{in}})}(k)|k, \underline{0}_{\text{in}}\rangle + \sum_{b=0}^{1} T_{\underline{b},\text{out}}^{(\underline{1},\text{in})}(k)|k, \underline{b}_{\text{in}}\rangle. \tag{4.1b}$$

These transformations are captured by the first two columns of the S matrix for the scattering process, defined as

$$S \doteq \begin{pmatrix} R_{\underline{0},\text{in}} & T_{\underline{0},\text{in}}^{(\underline{1},\text{in})} & T_{\underline{0},\text{in}}^{(\underline{0},\text{out})} & T_{\underline{0},\text{in}}^{(\underline{1},\text{out})} \\ T_{\underline{1},\text{in}}^{(\underline{0},\text{in})} & R_{\underline{1},\text{in}} & T_{\underline{1},\text{in}}^{(\underline{0},\text{out})} & T_{\underline{1},\text{in}}^{(\underline{1},\text{out})} \\ T_{\underline{0},\text{out}}^{(\underline{0},\text{in})} & T_{\underline{0},\text{out}}^{(\underline{1},\text{in})} & R_{\underline{0},\text{out}} & T_{\underline{0},\text{out}}^{(\underline{1},\text{out})} \\ T_{\underline{1},\text{out}}^{(\underline{0},\text{in})} & T_{\underline{1},\text{out}}^{(\underline{1},\text{in})} & T_{\underline{1},\text{out}}^{(\underline{0},\text{out})} & R_{\underline{1},\text{out}} \end{pmatrix} \tag{4.2}$$

in Reference [61], where it is also shown to be unitary.

The computational scheme dictates that a walker propagating toward the graph along

the input tails scatters such that at a later time it is propagating away from G along the output rails. To achieve this, it must be the case that

$$R_{\underline{0},\mathrm{in}}(k) \stackrel{!}{=} R_{\underline{1},\mathrm{in}}(k) \stackrel{!}{=} T^{(\underline{0},\mathrm{in})}_{\underline{1},\mathrm{in}}(k) \stackrel{!}{=} T^{(\underline{1},\mathrm{in})}_{\underline{0},\mathrm{in}}(k) \stackrel{!}{=} 0. \tag{4.3}$$

That is, there is neither reflection along an input rail nor transmission from one input to the other. When the conditions of Equation (4.3) are met, the scattering equations (4.1) of incoming walkers simplify to

$$|k,\underline{0}_{\mathrm{in}}\rangle \mapsto T^{(\underline{0},\mathrm{in})}_{\underline{0},\mathrm{out}}|k,\underline{0}_{\mathrm{out}}\rangle + T^{(\underline{0},\mathrm{in})}_{\underline{1},\mathrm{out}}|k,\underline{1}_{\mathrm{out}}\rangle, \tag{4.4a}$$

$$|k,\underline{1}_{\mathrm{in}}\rangle \mapsto T^{(\underline{1},\mathrm{in})}_{\underline{0},\mathrm{out}}|k,\underline{0}_{\mathrm{out}}\rangle + T^{(\underline{1},\mathrm{in})}_{\underline{1},\mathrm{out}}|k,\underline{1}_{\mathrm{out}}\rangle, \tag{4.4b}$$

so that a walker approaching the graph with momentum k in an arbitrary superposition of the two input tails,

$$|\psi_{\mathrm{in}}\rangle = \alpha|k,\underline{0}_{\mathrm{in}}\rangle + \beta|k,\underline{1}_{\mathrm{in}}\rangle, \tag{4.5}$$

is transformed to a superposition on the output tails by the operator

$$W \doteq \begin{pmatrix} T^{(\underline{0},\mathrm{in})}_{\underline{0},\mathrm{out}} & T^{(\underline{1},\mathrm{in})}_{\underline{0},\mathrm{out}} \\ T^{(\underline{0},\mathrm{in})}_{\underline{1},\mathrm{out}} & T^{(\underline{1},\mathrm{in})}_{\underline{1},\mathrm{out}} \end{pmatrix}. \tag{4.6}$$

This matrix is the lower-left block of S, which for the case in which Equation (4.3) holds can be written as

$$\begin{pmatrix} 0 & 0 & T^{(\underline{0},\mathrm{out})}_{\underline{0},\mathrm{in}} & T^{(\underline{1},\mathrm{out})}_{\underline{0},\mathrm{in}} \\ 0 & 0 & T^{(\underline{0},\mathrm{out})}_{\underline{1},\mathrm{in}} & T^{(\underline{1},\mathrm{out})}_{\underline{1},\mathrm{in}} \\ T^{(\underline{0},\mathrm{in})}_{\underline{0},\mathrm{out}} & T^{(\underline{1},\mathrm{in})}_{\underline{0},\mathrm{out}} & R_{\underline{0},\mathrm{out}} & T^{(\underline{1},\mathrm{out})}_{\underline{0},\mathrm{out}} \\ T^{(\underline{0},\mathrm{in})}_{\underline{1},\mathrm{out}} & T^{(\underline{1},\mathrm{in})}_{\underline{1},\mathrm{out}} & T^{(\underline{0},\mathrm{out})}_{\underline{1},\mathrm{out}} & R_{\underline{1},\mathrm{out}} \end{pmatrix} \doteq \begin{pmatrix} 0 & B \\ W & C \end{pmatrix}. \tag{4.7}$$

Since this is merely a special case of the general S matrix it must still be unitary, and therefore

$$S^\dagger S = \begin{pmatrix} 0 & W^\dagger \\ B^\dagger & C^\dagger \end{pmatrix} \begin{pmatrix} 0 & B \\ W & C \end{pmatrix} = \begin{pmatrix} W^\dagger W & W^\dagger C \\ C^\dagger W & B^\dagger B + C^\dagger C \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} I_2 & 0 \\ 0 & I_2 \end{pmatrix}, \tag{4.8}$$

from which it is immediately seen that W is unitary. Thus an arbitrary input state of the form in Equation (4.5) with $|\alpha|^2 + |\beta|^2 = 1$ is transformed by the presence of the widget graph G to the output state

$$|\psi_{\text{out}}\rangle = W|\psi_{\text{in}}\rangle, \tag{4.9}$$

where the action of W can be interpreted as a single-qubit unitary gate on the encoded computational space. The goal then, in the search for computational unitaries effected by graph scattering, is to identify those graphs for which the conditions of Equation (4.3) are satisfied at particular k values, and for each determine the corresponding unitary W.

4.1.1 Enumerating graphs

The number of non-isomorphic graphs on N vertices does not have a simple closed-form solution in general, but the quantities are well catalogued for values of N sufficient for our purposes [65]. The first few terms, for $N = 1$ to $N = 11$, are

$$1,\ 2,\ 4,\ 11,\ 34,\ 156,\ 1044,\ 12\,346,\ 274\,668,\ 1\,2005\,168,\ 1\,018\,997\,864. \tag{4.10}$$

The growth is super-exponential, and it is for this reason that the search was truncated after graphs of $N = 9$ vertices.

The 'nauty' (*no* *aut*omorphisms, *y*es?) suite of graph-theoretic software tools and libraries [66] includes the utility 'geng' (*gen*erate *g*raphs), which is capable of efficiently

producing the adjacency matrices for all non-isomorphic simple graphs on N vertices. The software routines we prepared proceed as follows. Using geng, a list of the 288 266 graphs on nine or fewer vertices is prepared. For each graph, for each way to attach four tails to it, and for each of the nine distinct momentum values $k = (p/q)\pi$ with $p \in \{2, 3, 4, 5\}$ and $q \in \{1, \ldots, p-1\}$, all partitions of the tails into two input and two output are tested to see if the conditions of Equation (4.3) are satisfied to numerical precision. Specifically, the conditions are taken to be satisfied if the sum of the magnitudes of the transmission coefficients to the output tails is at least $1 - 10^{-11}$. If they are, then that set of graph, momentum, and tail attachments becomes a candidate single-qubit gate. These tests involve the numerical construction of Equation (3.11) and its solution for the coefficients of the vector $|k, vm\rangle^G$ in order to populate the gate matrix W using Equation (3.12), and are accomplished with the GNU Scientific Library [67] and LAPACK [68] mathematical programming libraries, integrated with our custom code. A candidate unitary is discarded if its transformation matrix W has already been found at the same momentum for a previously tested widget of the same effective length, though a count is kept for each computational unitary operation of how many configurations produce it.

4.1.2 Comparing effective lengths

The final test to perform is to calculate the effective lengths for all paths through each candidate graph, and record the configuration as a valid single-qubit gate only if they are compatible. As described briefly in the description of Childs' model of computation in Section 3.3.1, there are two distinct scenarios under which the lengths can be compatible to implement a gate. If all four entries of the gate operation W are non-zero, then there are paths from each input rail to both output rails, yielding a total of four paths through the widget graph. In this case, the four lengths must be equal. Alternatively, either the diagonal or off-diagonal entries can vanish, in which case only the two effective lengths

derived from the non-zero entries of W must be equal. It is important to treat these cases independently, because when $T(k)$ vanishes for a path through the graph, the derivative of its argument at that momentum is irrelevant; there is no transmission. Thus requiring that this derivative agrees with those on paths for which there is transmission is too restrictive a condition.

For each candidate graph a nine-point finite-difference stencil is used to calculate the effective length of each path with non-zero transmission through the graph, implemented directly in C code with the complex-number data type defined by the GNU Scientific Library. If all calculated lengths are equal then the candidate graph indeed simulates a single-qubit gate, and its adjacency matrix is recorded along with the tail attachment configuration, momentum, W matrix, and effective length, as well as a unique record identifier.

4.1.3 Analysis

The total number of combinations of graph, tail attachments, and momentum under the parameters chosen is 1 262 489 148. Of these 1 960 316, or just over 0.15%, meet all of the requirements to simulate a single-qubit gate by graph scattering. Within this set of gates there is a great amount of redundancy, in that many single-qubit unitaries can be implemented at a given momentum and with a certain effective length by multiple non-isomorphic widget graphs. When this redundancy is taken into account, the number of distinct gates reduces substantially to 3380. A complete listing of these gates can be found in Table A.1 of Appendix A. While 3380 is a small percentage of the original number of combinations evaluated, the number of gates is seen to grow exponentially as a function of the number of vertices in the scattering graph, as shown by Figure 4.1.

Further analysis of the raw data involves their aggregation according to a variety of binning criteria. It is clear from symmetry arguments that the graph on one vertex with

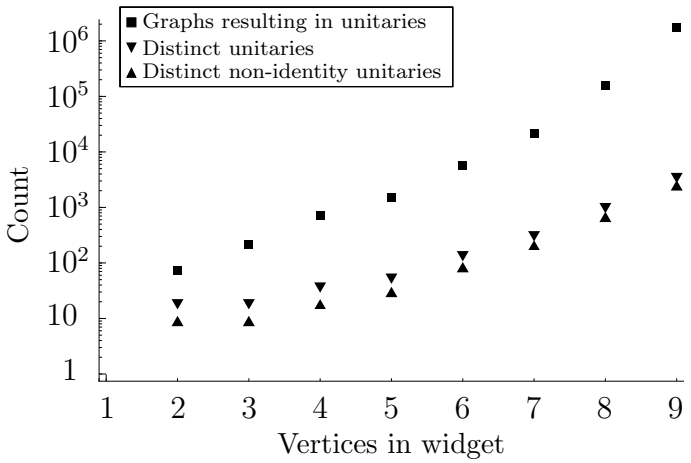


Figure 4.1: Total number of graphs resulting in unitary operators, as well as the number of distinct unitaries found as a function of number of vertices. Here operations are considered distinct if they arise from graphs with different effective lengths or at different momenta.

four tails attached to it cannot transfer a walker from one tail to only one or two of the remaining three and therefore no computational unitaries are expected on one vertex, and indeed none are found. Given two disconnected vertices it becomes trivial to implement either of the identity and X gates by attaching two tails to each vertex; if the two vertices are connected to each other then it turns out that no combination of attachment points allows for zero reflection along an incoming rail at any momentum, so no additional gates on two vertices are possible. These two gates on widgets of two disconnected vertices are shown in Figure 4.2 along with a third gate that can be implemented on five vertices, which is the smallest widget size that admits an operation other than I or X.

This shows that even after grouping together widgets according to the gate they implement, and the momentum and effective length at which they do so, there is still more redundancy within the set of gates identified, since there are exactly two possible gates on two vertices yet Figure 4.1 indicates that nearly 100 two-vertex gates were found. The exact number is 72, which is easily understood since each combination of graph, momentum, and tail attachments that results in a single-qubit gate is recorded.

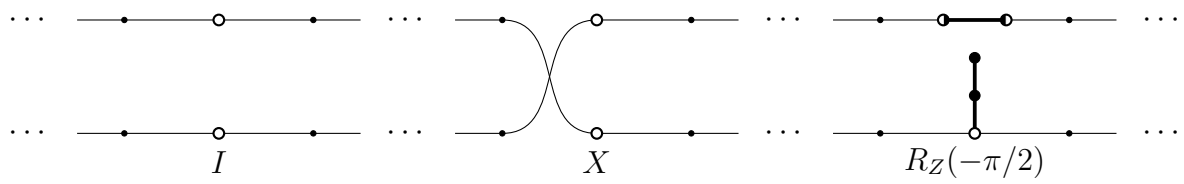


Figure 4.2: With only two vertices it is trivial to implement the identity gate I and the Pauli-X or SWAP operation at any momentum. The smallest number of vertices on which a gate other than I or X can be implemented is five, as for example in the widget shown here to implement $R_Z(-\pi/2)$ at momentum $k = \pi/2$.

The $n_G = 1$ graph of two disconnected vertices yields a gate at every momentum value, in particular at each of the $n_k = 9$ of them in our investigation. For each of the $n_a = 4$ partitions of four tails attachments into two pairs, the output tails can be labelled so as to implement one of the $n_u = 2$ unitaries I or X. Therefore the number of expected widgets to be identified should be $n_G n_k n_a n_u = 1 \times 9 \times 4 \times 2 = 72$, as it is. Clearly such redundancy is only likely to get worse as the number of vertices is increased—consider, for example, how many non-isomorphic graphs on nine vertices include two disconnected vertices that will exhibit exactly the same behaviour—so the primary task in performing the analysis of the generated data is to further aggregate the data so as to determine the actual growth in the number of available operations with widget size at each momentum. To this end, Figure 4.3 shows the counts of the set of distinct unitaries (points labelled '▾' in Figure 4.1) separated by momentum; in each case the increase is still exponential in the number of widget vertices.

4.2 Findings

Of the 3380 distinct combinations of W-matrix, momentum, and effective length, 2496 yield gates other than the identity. This of course means that there are $3380 - 2496 = 884$ different combinations of momentum and effective length at which the identity gate

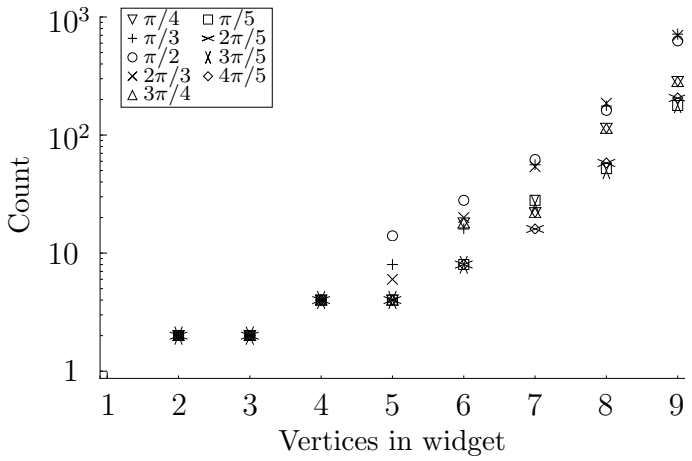


Figure 4.3: Number of graphs resulting in distinct unitaries, as presented in Figure 4.1, separated by the walker momentum.

can be simulated. This wide array of identities is far from trivial; in a multi-qubit quantum walk-based computation, the controlled version of a single-qubit unitary U can be implemented in a straightforward manner, but only if the identity can be performed at the same momentum and with the same effective length as the widget implementing U. While it is always trivial to implement an identity operation at any momentum for an integral effective length, widgets exist with non-integral effective lengths, both rational and irrational. Of the 2496 widgets that simulate operations other than the identity, 2234 have a commensurate identity widget and can thus be used to implement multi-qubit controlled versions of the same gates.

However, there is still redundancy in this gate set. There is no reason that two consecutive widgets must have the same effective length, only those in parallel. That is, if two widgets implement a given single-qubit unitary operation at momentum k, then they should not be considered distinct even if they have different effective lengths. Taking this into account reduces the total number of distinct unitaries to 332. Figure 4.4 shows the distribution of these gates at each momentum value investigated, from which it is clear that the number of gates is still an exponential function of the number of vertices

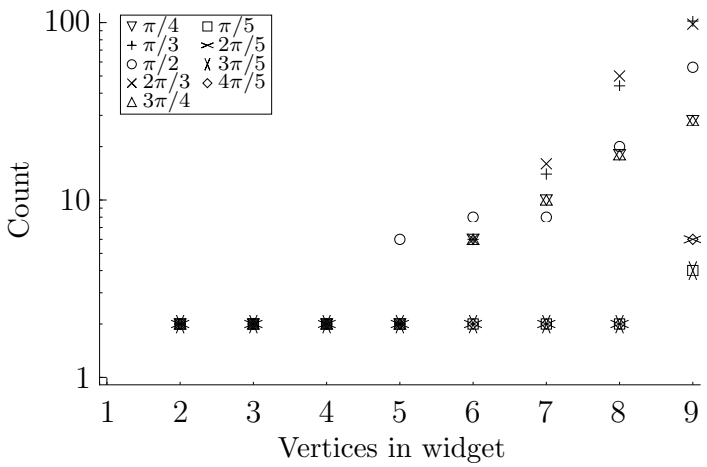


Figure 4.4: Number of distinct single-qubit gates available as a function of widget size. Single-qubit gates are considered distinct if they perform different computational unitary operations or if they are available at distinct momenta, but not if they differ only in their effective lengths.

for momenta that are integral multiples of $\pi/2$, $\pi/3$, and $\pi/4$. The small uptick at nine vertices for integral multiples of $\pi/5$ indicates that walkers at these momenta may not be incapable of computation, although the additional unitaries do not form a universal set. They are rotations of the Bloch sphere by irrational multiples of π, but are all X rotations so without a second axis of rotation cannot simulate arbitrary single-qubit operations.

At momentum $k = \pi/4$ the available gates include the Pauli-X, -Y, and -Z gates, as well as the identity operation and the universal set identified by Childs. In total 28 distinct single-qubit unitary operations can be simulated by a walker at $k = \pi/4$ with access to widgets on nine or fewer vertices. At $k = 3\pi/4$ there are also 28 unitaries, 23 of which are also found at $k = \pi/4$. With momentum $k = \pi/2$ the walker can simulate 56 rotations of the Bloch sphere about 32 non-parallel axes, while for $k = \pi/3$ and $k = 2\pi/3$ there are 102 and 98 rotations respectively, about 59 different axes in each case. The distribution of these axes of rotation across the surface of the Bloch sphere can be seen in Figure 4.5 for $k \in \{\pi/3, \pi/2, 3\pi/2\}$; the axes of rotation at other momenta all lie in either the equatorial plane of the Bloch sphere, or along its prime

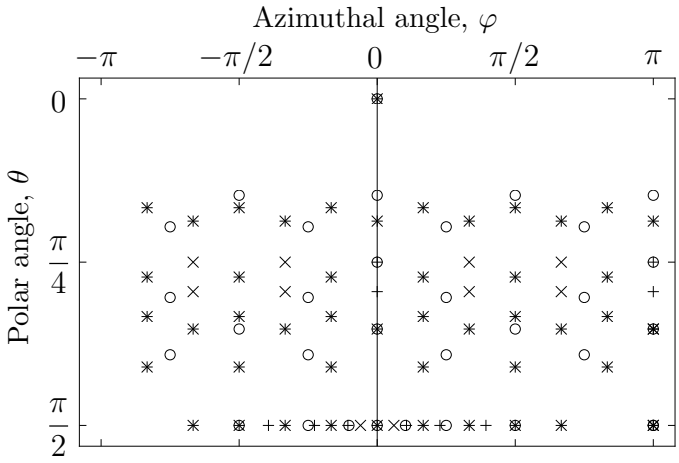


Figure 4.5: Distribution of the axes of rotation about which single-qubit unitaries can be effected at momenta $\pi/2$, $\pi/3$, and $2\pi/3$. That is, there exists at least one non-trivial rotation $R_{(\theta,\phi)}$ about each axis defined by these points (θ, ϕ) on the upper hemisphere of the Bloch sphere. Symbols are as in Figure 4.4.

meridian. The reflection symmetries about $\phi = 0$ and $\phi = \pm\pi/2$ are due to the different configurations of attachment points available for each graph. If a given widget performs a single-qubit rotation about the (θ, ϕ) axis of the Bloch sphere when the attachment vertices for $(|0\rangle_{\text{in}}, |1\rangle_{\text{in}}, |0\rangle_{\text{out}}, |1\rangle_{\text{out}})$ are $(|1\rangle, |2\rangle, |3\rangle, |4\rangle)$, then swapping the inputs and outputs by re-attaching the tails in the order $(|3\rangle, |4\rangle, |1\rangle, |2\rangle)$ results in a rotation about $(\theta, -\phi)$. Interchanging the inputs with each other and doing the same for the outputs, i.e. re-ordering the tails as $(|2\rangle, |1\rangle, |4\rangle, |3\rangle)$, leads to a rotation about $(\theta, \pi - \phi)$, which is equivalent to conjugating the original gate by X.

There are 16 unitaries that can be produced by graphs on five vertices, eight of which are 'new' in the sense that they cannot be produced by four or fewer vertices. Similarly, there are 24 new unitaries on six vertices, and 30 more on seven. These 62 unitary operators are produced by only 15 representative graphs, in concert with a variety of tail attachment and momentum configurations. One of these graphs yields gates at two variations of attachment points that lead to non-isomorphic infinite graphs once the tails

are included. This increases the number of distinct widget graphs to 16, all of which are shown in Figure 4.6.

4.2.1 Notable results

Figure 4.7 showcases four graphs that produce interesting and perhaps unexpected results. While there is no reason to assume *a priori* that this scattering formalism will not produce any rotations by irrational multiples of π for the configurations that are discussed, neither is it intuitive that this should be the case given that the graphs are scattering plane waves with momentum values that are rational fractions of π. Nevertheless, over 100 rotations through angles that numerically appear to be irrational fractions of π are identified. These individual results can be checked analytically by solving Equations (3.11) and (3.12) exactly in the cases of interest; those that were so checked bore out the apparent irrationality indicated by the numerical results. Figure 4.7(a) depicts one such case, a graph that implements the transformation $ZR_Z[\arctan(5\sqrt{3}/11)]$ up to a global phase at momentum $k = \pi/3$. Besides the novelty of obtaining irrational multiples of π under the circumstances, these graphs are inherently useful because any two rotations about non-parallel axes by irrational multiples of π form a universal set for single-qubit quantum computation [18].

Effective length is another feature of the widgets that provides some intriguing results. The momentum separator of Figure 3.7(b) has irrational effective lengths at each of the momenta it transmits perfectly, but it is only effective for a walker propagating on a single rail and as such cannot act as a widget in a computational graph. Graphs are identified that can act as single-qubit widgets, with either rational yet non-integral or irrational effective lengths from each input to each output. Figure 4.7(b) shows a graph on only five vertices that is capable of acting as an identity gate with an effective length of $\ell = 1/2$; at the momenta in question, no graph exists on fewer than five

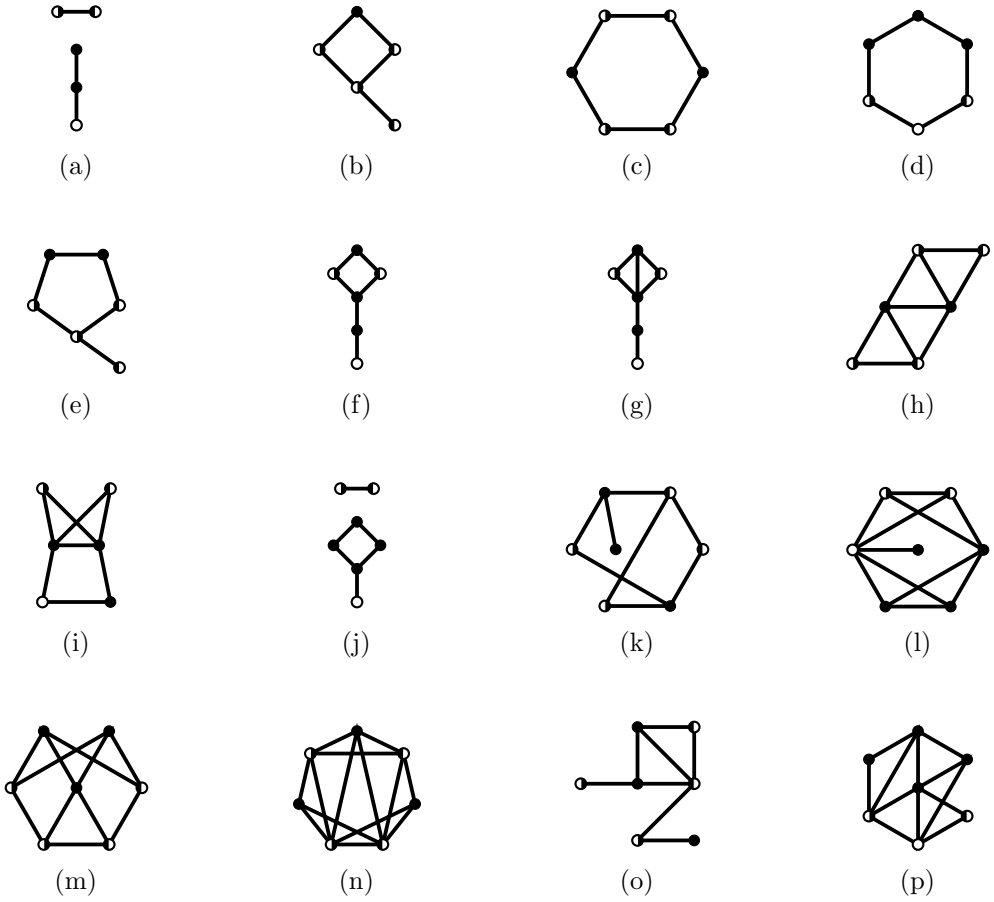


Figure 4.6: The graphs with five, six, or seven vertices that implement single-qubit unitaries which are unavailable on fewer vertices. The five-vertex graphs (a) and (b) result in eight distinct unitaries under different tail attachment configurations and different momentum values. The six-vertex graphs (c)-(i) yield a total of 24 unitaries, and the seven-vertex graphs (j)-(p) lead to 30. Note that while (c) and (d) are isomorphic, they represent two distinct, non-isomorphic configurations once tails are attached.

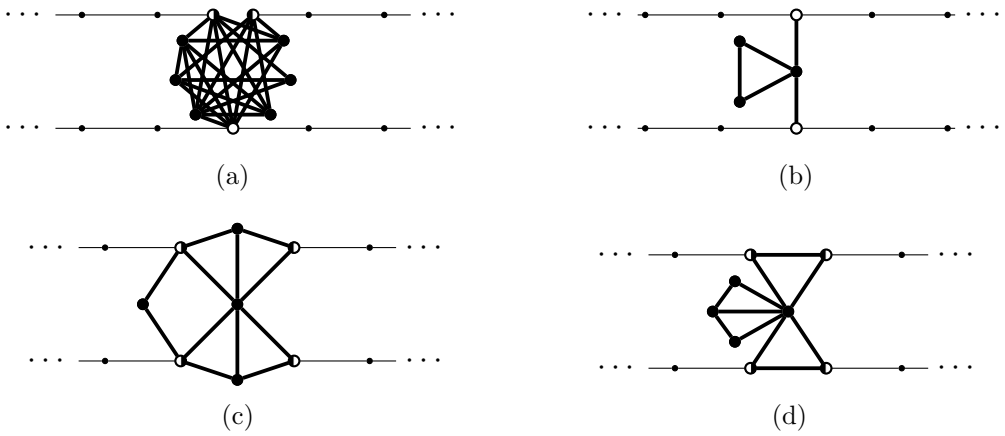


Figure 4.7: Some graphs that exhibit non-intuitive properties. At $k = \pi/3$, (a) performs a Z rotation through an angle of $\arctan\left(5\sqrt{3}/11\right)$, followed by a Z gate. Also at $k = \pi/3$, (b) is an identity gate with an effective length of $\ell = 1/2$. Finally at $k = \pi/4$, (c) performs the basis-changing operation $\sqrt{X}$ and (d) implements an identity, each with the irrational effective length of $\ell = 5 - 2\sqrt{2}$.

vertices that has a non-integral effective length and implements a single-qubit unitary. Finally Figure 4.7(c) shows a single-qubit widget that has an irrational effective length of $\ell = 5 - 2\sqrt{2}$. This graph implements the basis-changing operation $\sqrt{X}$, and remarkably the graph in Figure 4.7(d) acts as an identity gate with this same irrational length.

Of the 3380 graphs capable of implementing single-qubit unitaries, 2614 of them have non-integral effective lengths. Of these, 1072 appear numerically to have irrational lengths, including the 'longest' graph identified, which has effective length $\ell = 350 + 156\sqrt{5} \approx 698.826$. This extreme effective length due to a comparatively small number of vertices (i.e. $9 \ll 700$) is another observed phenomenon whose presence is not initially obvious. Such a length corresponds to the incoming wave packet's having been localized in the region of the graph for a significant duration, and is reminiscent of a diverging negative scattering length, approaching unitarity in traditional quantum scattering theory. Almost 20% of the unitaries identified have effective lengths $\ell \geq 10$, with greater than 1% having $\ell \geq 100$. In the same vein, it is also noteworthy that no widgets with negative effective

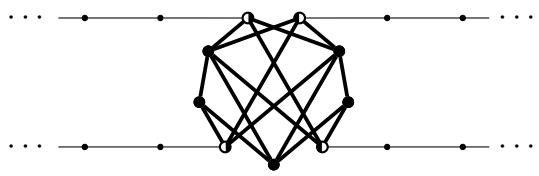


Figure 4.8: Graph that implements a Hadamard gate, up to an overall phase, at momentum $k = \pi/2$.

lengths were identified. As the derivative of the argument of the transmission coefficient there is nothing mathematically restricting the effective length to positive values, and indeed many graphs exhibit negative effective lengths over a range of momenta, but never at those corresponding to perfect transmission within the range of parameters in this study.

Additionally, Figure 4.8 shows a graph on nine vertices that implements a Hadamard operation, up to an overall phase, at momentum $k = \pi/2$. This graph as identified by our analysis was recently employed in a new proposal for universal quantum computation with multiple walkers [69].

4.3 Widgets as resources

In describing new models for quantum computation, an important consideration is always that of quantifying the associated resource costs, generally in terms of relatively abstracted concepts such as space, time, and energy, as opposed to specific objects such as beam splitters and transistors. While Childs makes it clear that he does not propose this single-walker scattering model as an experimental prescription for the implementation of a quantum computer, his work has nevertheless spawned numerous attempts to simulate quantum walk-based algorithms in laboratory situations [70–76]. Without such experiments in mind however, Childs presents an analysis in terms of the number of tails and computational wires required to implement a circuit containing m widgets

on n qubits. Specifically, he determines that $m_F = \log \Theta(m^2)$ filter widgets and tails of length $\Theta(m^2)$ are required. This in turn can be used to quantify the number of vertices required by the graph, which is a meaningful contribution to the resource requirements of the model since state preparation and final measurement are performed in the vertex basis.

Given a circuit containing m_1 single-qubit gates and m_2 two-qubit gates, the Solovay-Kitaev theorem, Theorem 1.2, guarantees that each of the single-qubit gates can be simulated with accuracy ε using at most $O(\log^c(1/\varepsilon))$ elements of the universal set of gates available as widgets, where c is a constant independent of ε. Corollary 1.1 states that each of the two-qubit gates can be likewise simulated with $O(\log^c(14/\varepsilon))$ widgets. This leads to a total of

$$m = O[m_1 \log^c(1/\varepsilon) + m_2 \log^c(14/\varepsilon)] \tag{4.11}$$

widgets on n qubits. Each single-qubit widget requires at most a constant $O(1)$ number of vertices when there is a single qubit to be simulated, but must be repeated 2^{n-1} times when n qubits are encoded; the two-qubit CNOT gate used to construct arbitrary two-qubit gates simply requires 2^n vertices and an appropriate rearrangement of the edges attaching them to the previous widget in the graph. Thus each of the m widgets requires $O(2^n)$ vertices, so the total number of vertices in the computational graph is

$$N_c = O(2^n m). \tag{4.12}$$

Additional vertices must be attached to the computational graph to form the 2×2^n input and output tails, and $m_F = \log \Theta(m^2)$ filter widgets, each containing an additional tail.

Each tail must be at least $O(m^2)$ vertices long, resulting in

$$N_t = O\left[(2^n + m_f)m^2\right] = O\left[(2^n + \log m)m^2\right] \tag{4.13}$$

further vertices in the tails. The total number of vertices in the graph as a whole is therefore

$$N_v = N_c + N_t = O\left[2^n(m + m^2) + m^2 \log m\right] = O\left[(2^n + \log m)m^2\right]. \tag{4.14}$$

The depth of the graph grows polynomially as a function of the number of gates to be simulated, which translates into an efficient runtime for the simulation since the walker simply traverses the polynomial number of widgets at a constant speed of $O(1)$. However, the exponential growth in the width of the graph with the number of qubits means that this model cannot be implemented directly as described in a scalable manner.

Chapter 5

Discontinuous quantum walks

The method of computing with a discrete-time quantum walker discussed in Section 3.4 and Reference [63] transfers a quantum walker from one widget to the next perfectly, and thus requires neither tails to support momentum eigenstates nor a high probability of needing to execute the computation multiple times before the walker returns an answer. In light of this and the discussion in Section 2.4 of perfect state transfer for continuous-time walkers, one might wonder whether these same results are possible for computing with a continuous-time quantum walk. In Reference [77] we show that they are, when an element of global control, akin to the flipping of a coin in the discrete-time case, is added to the walk. The resulting scheme is referred to as a discontinuous quantum walk, as it performs a quantum computation by subjecting a quantum walker to discrete steps of continuous evolution; the term is simultaneously a tongue-in-cheek portmanteau and an accurate description of the evolution of the walker, which occurs over a set of intervals on a graph with temporally piecewise-constant but discontinuous edge weights.

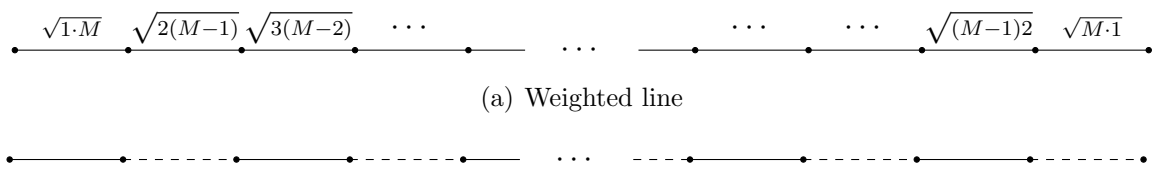
5.1 Perfect state transfer on an unweighted line

Perfect state transfer and its generalization to transfer between subsets of vertices are useful properties from the point of view of quantum information processing. With appropriately designed graphs, such as the NAND trees of Section 3.1.2, a walker can be transported such that its arrival in a particular location on the graph encodes the result of a computation. This arrival occurs at set intervals when the state of the walker is restricted to a well-defined subset of vertices, though in general throughout its evo-

lution the state of the walker is spread out over a larger portion—possibly all—of the graph. Though it is not phrased in this manner, the scheme for universal computation by continuous-time quantum walk of Reference [62] and Section 3.3 conceptually involves the transfer of a walker from its initial state on the subset of vertices corresponding to the $|\underline{0}\rangle^{\otimes n}$ input tail on one side of the computational graph, to the subset of vertices of the 2^n output tails on the opposite end of the graph, though in this case the transfer is not perfect. In the idealized version with semi-infinite tails and a walker in a momentum eigenstate there is of course no actual propagation or transport, and when the graph is made finite there exist components of the walker carried away by the momentum filters as well as reflected back along the input tails or internally within the computational graph. Nevertheless, the concept of perfect state transfer from one subset of vertices to another provides an excellent model for the goal of the scheme.

Perfect state transfer between subsets of vertices is also an obvious component of the discrete-time quantum walk scheme of Reference [63] and Section 3.4, in which the computational rails are designed specifically to localize the walker after s steps of the walk on those vertices a distance s from its initial location. As discussed in Section 3.1.2, PST appears in specific quantum walk-based algorithms such as that in Reference [60] for evaluating NAND trees, in which the initial state of the walker is localized to the vertices of the input tail, and the final state is localized on the input/output rail as a whole, with no population remaining in the computational graph despite the walker's having traversed it at intermediate times. As in Childs' scheme, when the tails are truncated to a finite length the transfer is not perfect, but conceptually the idea of perfect state transfer is still present.

The weighted line of M segments in Figure 5.1(a) transfers a continuous-time walker perfectly from one end to the other, but unlike the discrete-time version of the rail the walker is not localized on any vertex during its journey along the line. This makes the



(a) Weighted line

(b) Unweighted line of alternating graph segments

Figure 5.1: Two methods of perfect state transfer through $M+1$ vertices on a line. (a) A linear graph with always-on edges of appropriately tuned weights transfers a walker between the left-most and right-most vertices in time $\pi/2$. (b) If the edges are unweighted but can be turned on and off in an alternating fashion, the walker transfers from each vertex to the next in time $\pi/2$, traversing the entire line of M segments in time $M\pi/2$.

prospect of finding widgets to successfully transform the state of a localized continuous-time quantum walker in a computationally meaningful manner as it traverses a computational rail daunting. Furthermore, concatenating two graphs that each exhibit perfect state transfer does not in general result in a graph that also exhibits PST. This is made clear by the path graph, since a walker on P_2 can undergo PST from one vertex to the other, but while two concatenated copies of P_2—i.e. P_3—exhibits perfect state transfer between its ends, adding on even one more segment results in a graph on which no PST occurs between any pair of vertices. Thus attempts to create a universal set of widgets for a continuous-time quantum walk to be able to simulate quantum computations akin to the discrete-time model must use a different strategy.

There exists a straightforward adaptation to the line however under which the walker is still transferred perfectly yet also relocalizes on intermediate vertices before its traversal of the line is complete. Consider the two graphs — one with solid edges, one with dashed — in Figure 5.1(b). The vertices are fixed but the graphs can each be enabled or disabled by switching their edge weights between 0 and 1. If a walker is initially on one of the vertices and only one set of edges is enabled, then even though there are $M/2$ edges in the graph, the walker only 'sees' one of them so that its evolution is effectively governed by

K_2. It simply transfers back and forth between the initial vertex and the one connected to it [cf. Equations (2.31) and (2.32)]. This switching between graphs to affect the evolution of the walker in a sequence of steps is directly analogous to the interleaving of coin and shift operations in the discrete-time quantum walk model, but with an explicit Hamiltonian defined to generate each of the unitary operations and with no requirement for a coin degree of freedom.

5.1.1 State transformation by perfect state transfer

For a positive integer $n \geq 1$, consider a set V of at least 2^n vertices, some subset of which has been uniquely assigned to each of the integers from 0 to 2^{n-1}. That is

$$V = V_0 \cup V_1 \cup \cdots \cup V_{2^{n-1}} \cup V_\perp, \tag{5.1}$$

where the individual subsets V_i are non-empty and pairwise disjoint; $V_\perp$ may or may not be empty, but is disjoint from the other subsets. One can then define a quantum walk on a graph on V such that if a measurement in the vertex basis locates the walker on a vertex in V_i, then the walk is to be interpreted as having encoded the computational basis state given by the n-bit binary expansion of i. If the walker is found on a vertex in $V_\perp$ then no computational state is encoded. The goal will be to determine graphs on V such that the evolution of a walker effects a computational operation. For example, in the graph-scattering theory of the previous chapter $n = 1$ and the two computational-vertex sets are those of the rails,

$$V_b = \{x_{\underline{b}_{\text{in}}}\}_{x=1}^{\infty} \cup \{x_{\underline{b}_{\text{out}}}\}_{x=1}^{\infty} \tag{5.2}$$

for the two bit values $b \in \{0, 1\}$. The vertices of the widget constitute $V_\perp$.

Given such a collection of vertices, let $V_G \subseteq V_\perp$ and suppose that there exist i and

j such that $\{u\} \subseteq V_i$ and $\{w\} \subseteq V_j$. Consider a finite, weighted, undirected graph $G(V_G, E, w)$ on the vertex set $V_G = \{v_i\}_{i=1}^{N}$, with adjacency matrix A_G. Let $|v_u\rangle$ and $|v_w\rangle$, not necessarily distinct, be two vertices of G to which $|u\rangle$ and $|w\rangle$, respectively, have been attached. This attachment creates a new larger graph with adjacency matrix

$$A_{G+E} \doteq A_G + (|u\rangle\langle v_u| + |w\rangle\langle v_w| + \text{H.c.}), \tag{5.3}$$

where '$G+E$' is a shorthand notation representing the graph G with two edges attached. There are two distinct possibilities when this is the case: either u and w are part of the same set so that $i = j$, or they are not. First, suppose these vertices both encode the same computational basis state. Then if the graph G is such that there is perfect state transfer from u to w in time t_1, a walker initially on vertex u at time $t = 0$ will evolve into the state

$$|\psi(t_1)\rangle = \exp(\mathrm{i}At_h)\,|u\rangle = \mathrm{e}^{\mathrm{i}\varphi}|w\rangle \tag{5.4}$$

for some real φ. If a different graph similarly defined also exhibits PST between its two additional vertices in the same time but yielding a different phase factor on the resulting state, then these two graphs can simulate the action of a phase gate on a single qubit, identified by the single bit at which the binary expansion of i differs from that of the set containing the computational vertices of the second graph.

Alternatively, suppose that the two vertices u and w each encode a different computational basis state and the graph G in this case is such that the subset of vertices $\{u, w\}$ is periodic in time t_2. That is, the graph exhibits perfect state transfer from this subset back onto itself. Then a single-qubit rotation can be said to have been simulated. Given a normalized initial state $|\psi_0\rangle = \alpha|u\rangle + \beta|w\rangle$ and a time-evolved state

$$|\psi(t)\rangle = \mathrm{e}^{\mathrm{i}At}\,(\alpha|u\rangle + \beta|w\rangle), \tag{5.5}$$

such that $\langle v_i|\psi(t_2)\rangle = 0$ for each vertex $|v_i\rangle$ of G, it must also be the case that $|\psi(t_2)\rangle = \gamma|u\rangle + \delta|w\rangle$ for some γ and δ such that $|\gamma|^2 + |\delta|^2 = 1$, and therefore the state of the walker at this later time once again encodes a computational state, having undergone the transformation to $\gamma|u\rangle + \delta|w\rangle$.

Under these scenarios, the graph G is called a widget graph, and the two edges connecting $|u\rangle$ and $|w\rangle$ to G are referred to as a transport graph. These ideas form the basis of the discontinuous quantum walk scheme for universal computation; they are made explicit and given concrete examples in the following sections.

5.2 Hybridizing discrete- and continuous-time computation

With two parallel copies of the alternating two-graph line it becomes possible to encode a logical qubit, with each line acting as a computational 'transport' rail. Small widget graphs can be interspersed along and between the two rails as in Figure 5.2, and if each widget also exhibits PST but transforms the state of the walker either by adding a phase or transferring it from a single input vertex to a superposition of two output vertices then an encoded computational operation can be performed. In this manner, a quantum walker taking discrete steps of continuous evolution can simulate universal quantum computation. To affect the relative phases of the encoded $|\underline{0}\rangle$ and $|\underline{1}\rangle$ states, equivalent to a rotation $R_Z(\theta)$ of the qubit by an angle θ about the Z axis, one needs to add an identity widget to the first rail and a phase widget to the second. A continuous-time quantum walker must take the same amount of time to traverse each of these widgets, which must exhibit PST. To create a universal single-qubit gate set one also requires a rotation about an orthogonal axis X or Y. This requires a widget that connects the two rails, in such a way that after continuous evolution for a specified time the initial superposition of the amplitude on the two rails is transferred to a different superposition

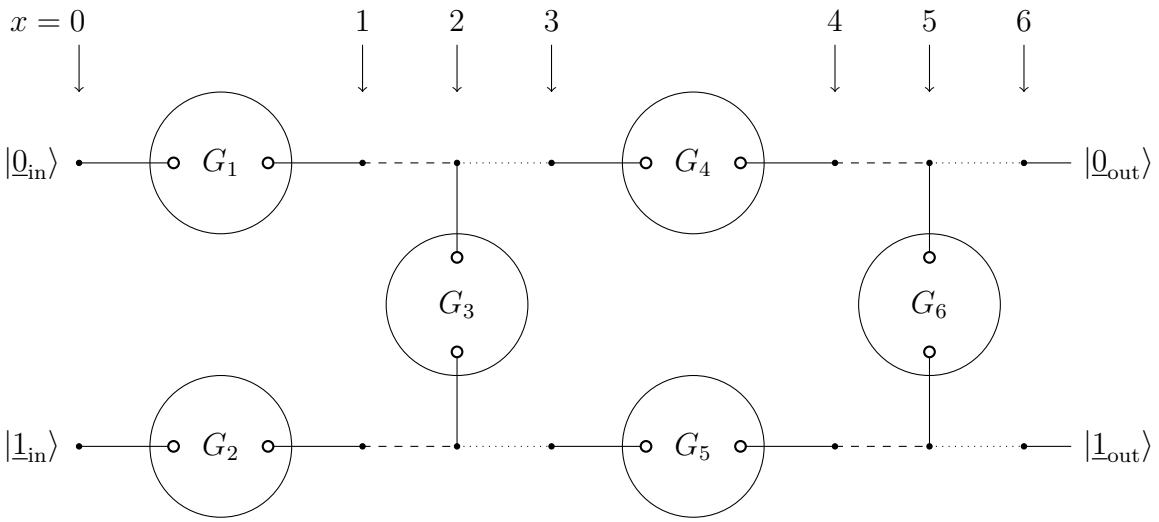


Figure 5.2: Layout of scheme for a discontinuous quantum walk on one qubit. The transport graphs $G_T^{(1)}$ (solid edges), $G_T^{(2)}$ (dotted), and $G_T^{(3)}$ (dashed) are each simple unconnected graphs over a single vertex set. The computational graph G_C, containing the elements G_i, is constructed from a universal set of widgets. The enabling of the transport graphs in a prescribed sequence causes a walker initially in a superposition of the vertices at $x = 0$ to undergo PST to $x = 6$, arriving in a transformed superposition. Open-circle vertices represent those to which the widgets G_i attach.

of $|\underline{0}\rangle$ and $|\underline{1}\rangle$, under the generalized version of PST in which a walker on a subset of vertices is transferred perfectly to another subset, occupying intervening vertices at intermediate time but leaving no amplitude outside the final subset at the end-time of the evolution.

Figure 5.2 demonstrates all of these elements combined to form a single-qubit gate via hybrid discrete-continuous quantum walk. The circled G_i elements indicate widget graphs to be attached to the network of transport rails; they are chosen from a universal set of widget graphs, one example of which is presented in Section 5.3, and comprise the computational graph G_C. The construction of G_C is algorithm dependent, but once specified it remains in place throughout the computational protocol, which requires a level of global control only to switch among the three sets of transport segments in a

prescribed sequence such that at most one is enabled at any given time.

5.2.1 Protocol for discontinuous computation

A walker is initialized on the left-most vertex of the $|\underline{0}\rangle$ rail at $x = 0$, and the solid transport graph $G_T^{(1)}$ is enabled. Note that at any given time only one transport graph — $G_T^{(1)}$, $G_T^{(3)}$, or $G_T^{(2)}$ — is enabled, so when one is stated to be on it is implied that the others are off. The widgets G_i attached to the horizontal transport rail segments are designed such that after a time t_h, the walker has transferred with unit probability to $x = 1$. The next step is taken with $G_T^{(3)}$ enabled for a time $t_m \doteq \pi/2$, moving the walker across the subgraph K_2 to $x = 2$ [cf. Equations (2.31) and (2.32)]. The graphs G_i for the vertical rail segments are such that after a time t_v, with $G_T^{(1)}$ enabled again, either the state of the walker remains unchanged or is transformed into a superposition of the $|\underline{0}\rangle$ and $|\underline{1}\rangle$ rails at $x = 2$. In either case $G_T^{(2)}$ is the next to be enabled, again for time t_m, moving the walker to $x = 3$, possibly in a superposition of the two rails. This sequence now repeats: $G_T^{(1)}$ for t_h, $G_T^{(3)}$ for t_m, $G_T^{(1)}$ for t_v, $G_T^{(2)}$ for t_m. Each iteration moves the walker three x positions to the right in a time of $t_h + t_v + \pi$, enacting operations upon it along the way. After traversing the whole graph, involving some number of iterations, the state of the walker at the output on the right will encode the desired arbitrary single-qubit state $\alpha|0\rangle + \beta|1\rangle$, with $|\alpha|^2 + |\beta|^2 = 1$. The next step is to expand this scheme from single-qubit operations to universal quantum computation, which follows from universal single-qubit computation plus a two-qubit entangling gate.

The simulation of two qubits requires four computational rails, and additional vertical connections among them. The number of transport graphs required does not change, and indeed is independent of the number of qubits to be simulated, though the sequence in which they are enabled is altered slightly to accommodate the additional connections between rails. Figure 5.3 illustrates the setup for the simulation of two qubits. In general

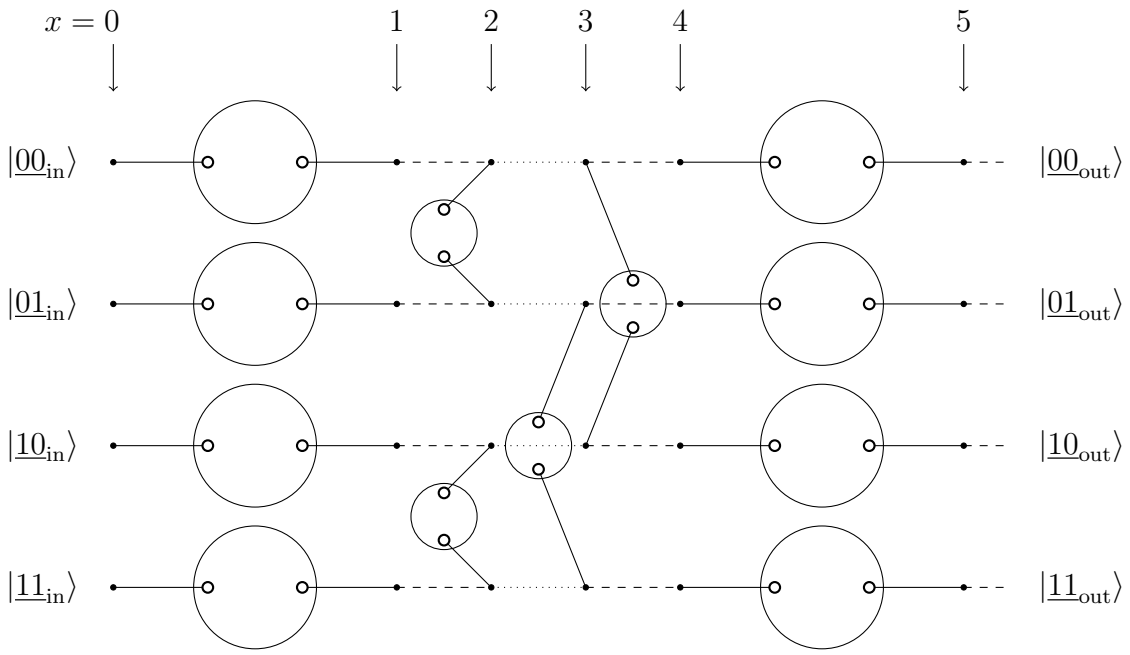


Figure 5.3: Extension of the discontinuous quantum walk scheme to two qubits. The horizontal portion of the protocol remains unchanged, and a second column of widgets is added to the vertical sequence; in general there are n such columns for the simulation of n qubits.

to simulated n qubits, the number of horizontal computational rails is 2^n as in the continuous-time scheme, though also as in that scheme this affects only the width of the graph and not its depth; the number of positions at which vertical inter-rail connections are required is only n for each iteration of the protocol. This is the case even though the simulation of a single-qubit gate on qubit i out of n requires a connection between rails $|\underline{b}_1 \cdots \underline{0}_i \cdots \underline{b}_n\rangle$ and $|\underline{b}_1 \cdots \underline{1}_i \cdots \underline{b}_n\rangle$ for all 2^{n-1} combinations of the bit values b_j, $j \neq i$, because those connections happen concurrently at the same position x.

There are three distinct step types that can be taken by the discontinuous quantum walker. A horizontal step $\mathcal{S}_h$ is taken when transport graph $G_T^{(1)}$ is enabled for a time t_h, followed by $G_T^{(3)}$ for a duration t_m. A vertical step $\mathcal{S}_v^{(i)}$ consists in the walker's evolving with the transport graph $G_T^{(1)}$ enabled for a time t_v followed by the graph $G_T^{(i)}$ for t_m. For example, in Figures 5.3, $\mathcal{S}_h$ transports the walker from $x = 0$ to $x = 2$, $\mathcal{S}_v^{(2)}$ evolves it through the vertical inter-rail links at $x = 2$ and then moves it to $x = 3$, where $\mathcal{S}_v^{(3)}$ propagates it to $x = 4$. This sequence now repeats to move the walker through the second sets of horizontal and vertical widget graphs. When n qubits are to be simulated, the order of steps required to move the walker from the beginning of ones set of horizontal gates to the beginning of the next is

$$\mathcal{S} \doteq \mathcal{S}_h, \underbrace{\mathcal{S}_v^{(2)}, \mathcal{S}_v^{(3)}, \ldots, \mathcal{S}_v^{(i)}}_{n \text{ steps}}, \tag{5.6}$$

where the final step $\mathcal{S}_v^{(i)}$ is of type $i = 2$ for an odd number of qubits, or $i = 3$ for an even number. The total duration of this sequence is $t_h + t_m + n(t_v + t_m)$. A computation on n qubits proceeds as follows. A walker is initialized on the vertex at $x = 0$ on the $|\underline{00} \cdots \underline{0}\rangle$ rail and the three transport graphs are cycled according to the sequence of steps $\mathcal{S}$. After each iteration of $\mathcal{S}$ the state of the walker is in a superposition of the vertices at a position $x \equiv 0 \pmod{n+2}$; after all required steps, say m of them, have been taken,

a measurement in the vertex basis of the position of the walker identifies one of the 2^n vertices at position $x = m(n+2)$ which in turn yields the outcome of the simulated computation as an n-bit string.

If additional runs of the algorithm are required, for example to build up statistics of the output state, then they can be run almost in parallel with multiple walkers. Once the first walker has reached the input node to the second round at position $x = n+2$, a second walker can be started at the input node of the first round, $x = 0$. With no additional cost, the same sequence $\mathcal{S}$ of transport graphs moves both walkers through the computation simultaneously, neither affected by the other's presence. When the first walker reaches the set of final output vertices, it remains there at the final x position while the last set of transport rails that it traversed is off. During this time it can be measured and ejected from the system before those rails cycle on again. This prevents the first walker from moving backward into the graph toward the second one. The whole process can of course be repeated for further additional walkers. Furthermore, the dependence of the distance between walkers on the number of qubits can be removed, making the separation constant, if the widgets are chosen such that $t_h = t_v$ and the number of vertical steps in (5.6) is odd. This requires the simulation of an odd number of qubits, which can always be arranged by the addition of an extra qubit if needed. In such a case $\mathcal{S}_v^{(3)} = \mathcal{S}_h$ and the entire sequence $\mathcal{S}$ becomes simply $\frac{1}{2}(n+1)$ repetitions of the sequence $\mathcal{S}_h, \mathcal{S}_v^{(2)}$. Therefore the second walker can be initialized at $x = 0$ after the first of these shorter sequences, i.e. when the first walker reaches $x = 4$, independent of n. For example, with $n = 3$ qubits the order in which the graphs are enabled for two walkers is

$$\begin{array}{lll} \text{Walker 1} & \rightarrow & \overbrace{\mathcal{S}_h, \mathcal{S}_v^{(2)}, \underbrace{\mathcal{S}_h, \mathcal{S}_v^{(2)}}, \overbrace{\mathcal{S}_h, \mathcal{S}_v^{(2)}}, \cdots}^{\text{Round 2}} \\ \text{Steps:} & & \underbrace{\qquad\qquad\qquad\qquad}_{\text{Round 1}} \\ \text{Walker 2} & \rightarrow & \end{array}^{\text{Round 1}} . \tag{5.7}$$

The universal gate set described in the next section is of this form, with $t_h = t_v$.

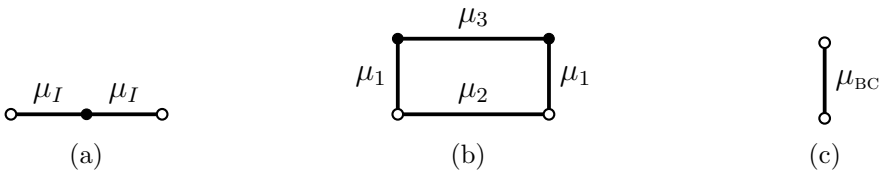


Figure 5.4: A set of widgets providing universal computation. (a) With $\mu_I \doteq \sqrt{3/2}$, this identity widget G_I turns a horizontal rail segment into an identity gate. (b) Edge weights $\mu_1 \doteq 5\sqrt{3}/8$, $\mu_2 \doteq 15/8$, and $\mu_3 \doteq 21/8$ allow this phase widget G_P to give the walker a phase factor of $\mathrm{e}^{-\mathrm{i}\pi/2}$ relative to what it acquires traversing the identity widget. (c) This single weighted segment, with $\mu_{\mathrm{BC}} \doteq 2\sqrt{3}$, splits the state of a walker on a computational rail into a superposition of both rails, effecting a change of basis.

5.3 A universal widget set

The set of widgets defined in this section contains graphs that, when inserted in a computational graph, can simulate phase gates and a basis-changing gate that together are efficiently universal for single-qubit computations. Depicted in Figure 5.4, their state-transfer properties are discussed here, followed by their application to single-qubit gates.

Consider first the two-edge weighted line in Figure 5.4(a) with edge weights $\mu_I \doteq \sqrt{3/2}$. When attached to a transport rail as in Equation (5.3), this widget becomes a reweighted version of the line in Figure 5.1(a), which therefore exhibits PST in an appropriately reweighted time. Specifically, instead of evolving under the adjacency matrix

$$A^{(4)}_{\mathrm{line}} \doteq \begin{pmatrix} 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & \sqrt{6} & 0 & 0 \\ 0 & \sqrt{6} & 0 & \sqrt{6} & 0 \\ 0 & 0 & \sqrt{6} & 0 & 2 \\ 0 & 0 & 0 & 2 & 0 \end{pmatrix} \tag{5.8}$$

for a time $\pi/2$ as it would to traverse the line with known PST, the walker evolves under

the Hamiltonian furnished by the adjacency matrix $A_{I+E} = \frac{1}{2}A^{(4)}_{\text{line}}$ for a time π. Since

$$\exp\left(\mathrm{i}A^{(4)}_{\text{line}}t\right) = \exp\left(\mathrm{i}A_{I+E}2t\right) \tag{5.9}$$

and the line defined in Equation (5.8) transfers the walker from one end to the other, so too does the widget graph G_I attached to the transport rails, described by A_{I+E}, but in twice the time. Since this widget may be inserted on a single computational rail, the resulting phase after PST maybe be computationally significant. The evolution between the attached vertices $|u\rangle$ and $|w\rangle$ is

$$\exp\left(\mathrm{i}A_{I+E}t_h\right)|u\rangle = |w\rangle, \tag{5.10}$$

where $t_h \doteq \pi$ has been defined based on the time taken for PST to occur.

The next widget graph is the weighted square G_P shown in Figure 5.4(b) which has adjacency matrix

$$A_P \doteq \begin{pmatrix} 0 & \mu_1 & 0 & \mu_2 \\ \mu_1 & 0 & \mu_3 & 0 \\ 0 & \mu_3 & 0 & \mu_1 \\ \mu_2 & 0 & \mu_1 & 0 \end{pmatrix}, \tag{5.11}$$

with edge weights $\mu_1 \doteq 5\sqrt{3}/8$, $\mu_2 \doteq 15/8$, and $\mu_3 \doteq 21/8$. Again labelling the two additional vertices as $|u\rangle$ and $|w\rangle$, attached to G_P at vertices $|v_1\rangle$ and $|v_4\rangle$ respectively, in the basis ordering of Equation (5.11), a walker on one attached vertex evolves to the other as

$$\exp\left(\mathrm{i}A_{P+E}t_h\right)|u\rangle = \mathrm{e}^{-\mathrm{i}\pi/2}|w\rangle. \tag{5.12}$$

Clearly then this widget attached to one computational rail can be combined with the widget G_I on the other computational rail to effect a phase gate in time t_h.

The third widget under consideration is the single weighted edge of Figure 5.4(c) which unlike the other two does not exhibit PST between the attached vertices; instead, when the edge weight is $\mu_{\text{BC}} \doteq 2\sqrt{3}$, a walker initially in an arbitrary superposition of the vertices $|u\rangle$ and $|w\rangle$ transfers in time $t_v \doteq \pi$ to a different superposition of those vertices, having zero overlap with the vertices of G_{BC}. That is, it undergoes a change of basis. A walker initially on vertex $|u\rangle$ evolves in time t_v according to

$$\exp\left(\mathrm{i}A_{\text{BC}+E}t_v\right)|u\rangle = \cos(\sqrt{3}\pi)|u\rangle + \mathrm{i}\sin(\sqrt{3}\pi)|w\rangle. \tag{5.13}$$

Due to the symmetry of the graph, it must be the case that a walker beginning its evolution on $|w\rangle$ evolves to the same state, but with $|u\rangle$ and $|w\rangle$ interchanged. That is, if $|u\rangle \equiv |\underline{0}\rangle$ and $|w\rangle \equiv |\underline{1}\rangle$ then the widget simulates the computational unitary

$$U_{\text{BC}} = \begin{pmatrix} \cos(\sqrt{3}\pi) & \mathrm{i}\sin(\sqrt{3}\pi) \\ \mathrm{i}\sin(\sqrt{3}\pi) & \cos(\sqrt{3}\pi) \end{pmatrix}, \tag{5.14}$$

which is equal to the X rotation $R_X(-2\sqrt{3}\pi)$.

Finally, there must be a method for implementing a single-qubit identity operation for the inter-rail vertical sections of the computational graph. This is in fact straightforward to implement by simply not inserting a widget at those positions in the graph (or equivalently, inserting the widget G_{BC} modified to have edge weight $\mu_{\text{BC}} = 0$). Since the specified evolution time $t_v = \pi$ and the single edge K_2 is periodic in time π, the missing edge causes a walker on either $|u\rangle$ or $|w\rangle$ to return to its initial state, up to an overall phase, in time t_v.

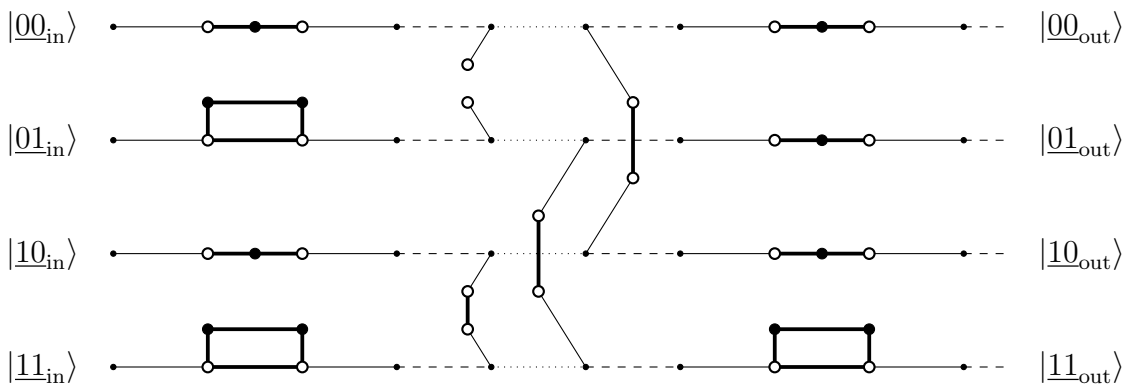


Figure 5.5: Example circuit executing four two-qubit gates. The first column of widgets applies a phase shift to $|\underline{01}\rangle$ and $|\underline{11}\rangle$ relative to $|\underline{00}\rangle$ and $|\underline{10}\rangle$, implementing the local gate $I \otimes \sqrt{Z}$. The second column performs an X rotation on the two states in which the first qubit is in state $|\underline{1}\rangle$, and does nothing to the other two, effecting a $\text{C}R_X$ gate on the pair of qubits. The third column implements $R_X \otimes I$, and the fourth, a CPHASE gate.

5.3.1 Constructing gates from widgets

These four widgets, those of Figure 5.4 plus the vertical identity, can be added to the transport rails in a variety of combinations, such as those demonstrated in Figure 5.5. In particular, if the horizontal identity widget G_I is attached to each rail encoding qubit i in state $|\underline{0}\rangle$, and the phase widget G_P to the remaining rails on which the same qubit is in state $|\underline{1}\rangle$, then Equations (5.10) and (5.12) show that the phase gate

$$\sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-\mathrm{i}\pi/2} \end{pmatrix} \tag{5.15}$$

can be simulated on that qubit. If the positions of the two widgets are swapped so that G_I applies to $|\underline{1}\rangle$ and G_P to $|\underline{0}\rangle$, then $\sqrt{Z}^\dagger$ is performed instead, up to an overall phase. Equation (5.14) already shows how a basis-changing R_X gate can be implemented when only one qubit is present. The same operation can be performed on qubit i out of n by inserting the widget G_{BC} between each of the pairs of rails such that all other qubits are equal. That is, the widget should join $|\underline{b}_1 \cdots \underline{0}_i \cdots \underline{b}_n\rangle$ and $|\underline{b}_1 \cdots \underline{1}_i \cdots \underline{b}_n\rangle$, for each of the

2^{n-1} bitstrings $b_1 \cdots b_{i-1} b_{i+1} \cdots b_n$.

A two-qubit entangling gate is also required, and the implementation of $\sqrt{Z}$ by the application of a phase to a single computational basis state suggests a straightforward method for the implementation of a controlled-$\sqrt{Z}$ gate, $\textsc{c}\sqrt{Z}$. With four rails encoding two qubits, simply attaching the phase widget G_P to the $|\underline{11}\rangle$ rail and the identity widget G_I to the $|\underline{00}\rangle$, $|\underline{01}\rangle$, and $|\underline{10}\rangle$ rails shifts the phase of the $|\underline{11}\rangle$ basis state by $\mathrm{e}^{-\mathrm{i}\pi/2}$ relative to the other three states, simulating the computational unitary

$$\textsc{c}\sqrt{Z} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\mathrm{i} \end{pmatrix}. \tag{5.16}$$

Repeating this obviously results in a full $\textsc{cz}$ operation. Furthermore, this method trivially extends to larger numbers of qubits, making the implementation of multiply-controlled operations no more difficult than that of local gates. Similarly, applying the basis-changing operation $U_{\textsc{bc}}$ between only a subset of the vertical rails on a given qubit allows for the implementation of controlled version of the R_X gate as well, as demonstrated in Figure 5.5.

5.4 Resource considerations

Up to unimportant overall phases, the single-qubit gate set described thus far for the discontinuous quantum walk model of computation is

$$\left\{ I, \sqrt{Z}, \sqrt{Z}^{\dagger}, T_X \right\}, \tag{5.17}$$

where $T_l \doteq R_l(-2\sqrt{3}\pi)$. This set also generates

$$T_Y = \sqrt{Z}^\dagger T_X \sqrt{Z}, \tag{5.18}$$

and as X and Y are non-parallel axes of the Bloch sphere and $-2\sqrt{3}\pi$ is an irrational multiple of π, Theorem 1.1 states that the set $\{T_X, T_Y\}$ is universal for single-qubit computation. To see that the available widgets generate a set that is efficiently universal, define matrices

$$A = \left(\sqrt{Z}\right)^2 T_X \left(\sqrt{Z}\right)^2, \tag{5.19a}$$

$$B = \sqrt{Z} T_X \sqrt{Z}^\dagger, \tag{5.19b}$$

and let

$$\mathcal{G} = \left\{\sqrt{Z}, \sqrt{Z}^\dagger, T_X, T_Y, A, B\right\}. \tag{5.20}$$

Then $\mathcal{G}$ is closed under inverse, since

$$\sqrt{Z}\sqrt{Z}^\dagger = \sqrt{Z}^\dagger\sqrt{Z} = I, \tag{5.21a}$$

$$AT_X = T_X A = I, \tag{5.21b}$$

$$BT_Y = T_Y B = I. \tag{5.21c}$$

Therefore $\mathcal{G}$ satisfies the conditions of Definition 1.3 and is an instruction set for SU(2). Thus the Solovay-Kitaev theorem, Theorem 1.2, guarantees that an arbitrary single-qubit unitary can be efficiently approximated to arbitrary accuracy by the gates of $\mathcal{G}$.

Consider as in Section 4.3 a given circuit to be implemented, containing m_1 single-qubit gates and m_2 two-qubit gates. As in the continuous-time case, each of the two-qubit gates can be simulated exactly with a constant number of the available entangling gates

and a constant number of arbitrary single-qubit gates. Thus the number of single-qubit gates that must be approximated to accuracy ε by elements of the set $\mathcal{G}$ is $m_g = m_1 + m_2$, and the number of gates required from the set is

$$m = O[m_1 \log^c(1/\varepsilon) + m_2 \log^c(14/\varepsilon)], \tag{5.22}$$

the same as in Equation (4.11). Each widget plus transport rail segments requires a constant number of vertices, repeated on up to 2^n rails for a total of $O(2^n)$ vertices per widget. Unlike in the continuous-time case, there are no additional vertices to be attached in tails; the total number of vertices in the graph as a whole is therefore

$$N_v = O(2^n m), \tag{5.23}$$

a polynomial savings in m over the continuous-time model. As in both previous models, the exponential size of the graph in this case appears only in its width, allowing the walker to cross the depth that is polynomial in the number of gates in a similarly scaled time.

5.5 Computational read-out

When the walker has traversed the computational graph, having transferred through the m widgets comprising the circuit and arrived at the final set of vertices at $x = x_f$, all three transport graphs are disabled. At this point, the position of the walker is in a superposition of the 2^n disconnected vertices at position x_f and will remain so as long as the edges of the transport graph $G_T^{(1)}$ are not present. A measurement of the walker in the vertex basis collapses its state onto a single vertex, revealing the bitstring that is the result of the computation. Alternatively, a measurement of $p \leq n$ qubits can be

accomplished either in a single step with a set of 2^p physical measurement operators, each spatially delocalized across 2^{n-p} vertices, or as a sequence of p single-qubit measurements, each requiring two measurement operators delocalized over 2^{n-1} vertices.

The discontinuous-walker scheme provides the best-case scenario of there being a one-to-one mapping between the computational basis states and a subset of 2^n vertices from the graph. For $z \in \{0, \ldots, 2^p - 1\}$, let $z_1 \cdots z_p$ be the p-bit binary expansion of z. To make a measurement of qubits $\{i_k\}_{k=1}^{p}$, let $\mathcal{V}_z$ be the set of integers between 0 and $2^n - 1$ whose n-bit binary expansions have bit value z_k at bit position i_k. Then the required measurement operators are

$$\left\{ P_z \doteq \sum_{v \in \mathcal{V}_z} |v\rangle\langle v| \right\}_{z=0}^{2^m-1} . \tag{5.24}$$

Clearly there are 2^p values of z, and for each z there are 2^{n-p} unspecified bit values so $|\mathcal{V}_z| = 2^{n-p}$. To implement p' single-qubit measurements, simply set $p = 1$ and repeat p' times. This exponential growth in the spatial extent of the required measurement operators is why the single-walker quantum-walk schemes for computation are proposed primarily in terms of computational capability, and not in terms of possible physical implementations. It also makes the prospect of implementing quantum error correcting codes in such systems impractical at best. With these considerations in mind, the next logical step is to consider the possibility of computing with multiple walkers. Doing so provides access to a tensor-product structure in the Hilbert space of the system, which can be easily exploited to encode qubits on an exponentially smaller graph than that required with only a single walker. This idea forms the basis of the next chapter.

5.6 Summary

The discontinuous quantum walk scheme shows how to harness phenomenon of perfect state transfer for universal quantum computation. With only one walker, the computational model is based on one rail per computational basis state, as developed in prior quantum walk schemes, both continuous and discrete. As in the discrete-time model, the discontinuous walker does not require the excess tails used in the continuous case to support well-defined momentum states, or momentum filters that prevent most of the walker from participating in the computation. The use of perfect state transfer ensures that the walker completes the quantum computation with certainty. Unlike in the discrete case, the Hamiltonian generating the time translation of the walker is clearly specified, and site-dependent coins of multiple dimensions are not required, nor indeed any coin at all. The cost associated with these modifications is an additional amount of global control, which is analogous to the coin and shift operations employed by discrete-time quantum walks with site-independent coins. The required control is algorithm independent, conforming to a well-defined, preprogrammed sequence.

The widgets described in Section 5.3 are efficiently universal for quantum computation, generating the instruction set $\mathcal{G}$ of Equation (5.20), so they provide a proof-of-principle scheme for the implementation of arbitrary quantum algorithms. However, they are neither unique nor are they likely to be a preferred set given some specific application. Alternative choices of single and double-rail graphs (generating single-qubit gates) might generate particular desired gates (such as the Hadamard or $\pi/8$ gate) more readily. Multi-qubit gates (such as the universal three qubit Toffoli gate) could be found by graphs linking multiple rails with widgets that exhibit the generalization of perfect state transfer to subsets of vertices. A desired unitary on n qubits would conceivably have a more efficient decomposition in terms of a larger widget set.

Chapter 6

Multiple walkers and the Bose–Hubbard model

The previous chapters have presented three distinct schemes for computing with quantum walkers, each of which requires a number of vertices exponential in the number of qubits to be simulated. This is not surprising, because a system of n qubits inhabits a Hilbert space spanned by 2^n computational basis states and a single quantum system, with no tensor product structure, is being used to address each of them uniquely. One straightforward adaptation that allows for the exponentially growing Hilbert space without necessitating a similarly sized graph is to employ multiple quantum walkers. As a step in that direction, a framework for the description of multiple non-interacting quantum walkers, distinguishable or not, has been put forth in the discrete-time case [78]. However, if there are multiple walkers on a graph and they do not interact then there is no meaningful difference from the situation of multiple distinct copies of the single-walker situation. Non-interacting walkers evolve independently of each other, so while they can be useful if many runs are required to build up statistics of the output state, no essentially new dynamics can be present. In the discrete-time quantum walk, effective interactions between otherwise non-interacting walkers can be induced through the sharing or swapping of coins [79], and entanglement can arise between a single walker and its coin [80]. Continuous-time quantum walks on the other hand have no coin degree of freedom, so they must rely on inter-walker interactions to introduce additional dynamics and generate entanglement.

There is evidence that on a given graph, multiple interacting continuous-time walkers are more computationally powerful than either a single walker or multiple non-interacting ones, when applied to the graph isomorphism problem [81], though it has been shown

that the number of such walkers must scale with the size of the graph [82, 83]. Since a single quantum walker is universal for quantum computation, given appropriate graphs on which to walk, it seems clear that additional walkers will not remove this power, and thus multiple walkers are also universal for quantum computation. However what is not immediately obvious is whether the added walkers can allow a multi-walker system to simulate the same computation as a single walker, but on an exponentially smaller number of vertices. In Reference [84] we show that indeed this is the case by presenting an explicit construction of an efficiently universal gate set based on multiple discontinuous quantum walkers, in which the number of walkers and size of the underlying graph are both linear in the number of qubits to be simulated; this model and an exposition thereof is the basis of the current chapter.

The Bose–Hubbard model, defined as it applies to quantum walks on graphs in the next section, also provides an excellent description of bosonic atoms confined to the sites of an optical lattice [85]. Such systems have been proposed as candidates for generating entanglement through two-qubit gates [86] and more generally for implementations of quantum computers [87], using continuously time-varying potentials in the adiabatic limit. A more recent review of these and similar schemes, with an emphasis on the proposed experimental implementations, can be found in Reference [88]. Due to the strong link between the Bose–Hubbard model and ultracold atomic systems there are a number of similarities between the content of these papers and the work in this chapter and Reference [84]; there are also several distinctions between these bodies of work.[1]

The most notable of these lies in the internal degrees of freedom required to encode

[1]It has come to my attention while editing this thesis, several months after the publication of my manuscript [84] on which this chapter is based, that Ionicioiu and Zanardi [89] had previously presented a scheme for quantum information processing under the Bose–Hubbard model that has significant overlap with some of the work in this chapter. There are minor cosmetic differences between our two proposals, but the qubit encoding and many of the gates are essentially identical and I regret not having become aware of their contribution earlier. However their focus is on cold trapped atoms, and the portions of this chapter that relate the scheme to quantum walks and perfect state transfer, and which describe quantum error correcting codes for multiple discontinuous walkers, remain distinct.

qubits in spin or energy states of the atoms: in quantum walk schemes for computation, position of the only degree of freedom available to the walkers, and is the mechanism used to encode quantum information. Additionally, the quantum-walk nature of the present work provides an important correspondence between the discontinuous evolution of interacting indistinguishable particles on polynomial-sized graphs, and that of a single discontinuous quantum walker on an exponentially larger graph. That is, a number of walkers linear in the number of qubits to be simulated n, walking on a vertex set of a size also linear in n, has the same power as a quantum walk with a constant number of walkers (in particular, one) on a graph with a number of vertices exponential in n.

6.1 Multiple walkers on a graph

A set of quantum walkers inhabiting a single graph is a particular case of a quantum many-body system on a generalized lattice. It is convenient to describe such systems with second-quantized notation, particularly for the case of indistinguishable walkers, and the graph-based nature of a quantum walk makes the Bose–Hubbard model an obvious choice for the description of multiple walkers on a graph. The Bose–Hubbard Hamiltonian as it applies here can be written for a weighted graph $G = (V, E, w)$ as

$$\mathcal{H}_G = -\sum_{\langle u,v\rangle} \tau_{uv} c_u^\dagger c_v + \frac{g}{2} \sum_v \mathcal{N}_v(\mathcal{N}_v - 1) + \sum_v \mu_v \mathcal{N}_v, \tag{6.1}$$

where the first sum runs over neighbouring vertices $\langle u, v\rangle$ such that $(u, v) \in E$ and the second runs over all vertices $v \in V$. The edge weight between u and v is $\tau_{uv} = \tau_{vu}$, which can be thought of as the hopping rate between the sites. The operator $c_v^\dagger$ creates a walker on vertex v, g is the on-site interaction strength between pairs of walkers occupying the same vertex, and $\mathcal{N}_v \doteq c_v^\dagger c_v$ is the number operator on vertex v. A local potential μ_v can be applied to each site, and it is assumed that $\mu_v = 0$ in general except when otherwise

specified.

In the case of a single walker either $\mathcal{N}_v$ or $\mathcal{N}_v - 1$ vanishes for every vertex with only a single walker present, and since the occupation basis $\{c_v^\dagger|\text{vac}\rangle\}_{v\in V}$ is isomorphic to the vertex basis $\{|v\rangle\}_{v\in V}$, the remaining sum is

$$\sum_{\langle u,v\rangle} \tau_{uv} c_u^\dagger c_v = \sum_{\langle u,v\rangle} \tau_{uv} |u\rangle\langle v| = A_G, \tag{6.2}$$

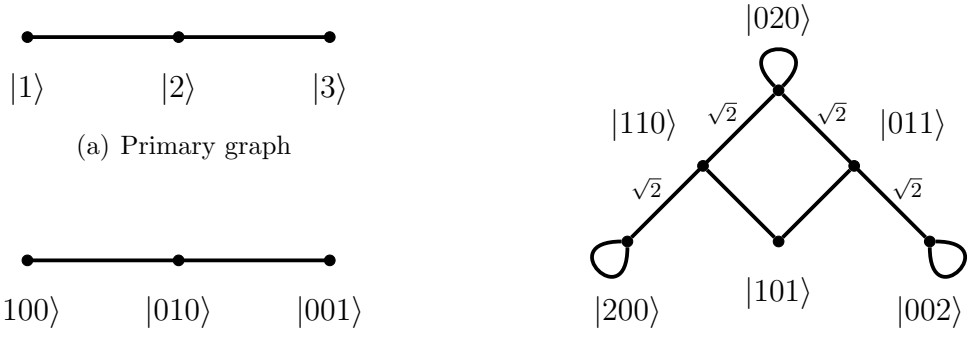
the adjacency matrix of G. Here

$$|\text{vac}\rangle \doteq |0_1 0_2 \cdots 0_v\rangle \tag{6.3}$$

is the vacuum state on the graph, in which no walkers are present and consequently all vertices are unoccupied. Thus Equation (6.1) recovers the expected description of a single quantum walker, $\mathcal{H} = -A$.

6.1.1 Primary and secondary graphs

The graph G on which the quantum walkers evolve, which defines the hopping rates τ_{uv} and yields the system Hamiltonian in the single-walker case, is the *primary graph* of the multi-walker system. When $M \geq 2$ walkers are present on a graph, one can also consider the *secondary graph* $G^{(M)}$, in which the Fock occupation-number states over G with a total occupation of M comprise the vertex set, and allowed transitions among them provide edges and their weights [90]. For example consider Figure 6.1, in which the path on three vertices P_3 is taken to be the primary graph, and the secondary graphs induced by one and two walkers upon it are shown. The vertex labels of the secondary graph are expressed in the second-quantized notation introduced in Section 2.5.1. Entry v of the ket specifies the number of walkers on vertex v in the primary graph; for example in Figure 6.1(c) the state $|110\rangle$ represents the presence of one walker on each of the first



(a) Primary graph

(b) Secondary graph with one walker

(c) Secondary graph with two walkers

Figure 6.1: Primary graph and corresponding secondary graphs induced by one and two walkers. (a) The primary graph P_2, to which an arbitrary number of walkers can be added, has three vertices. (b) With a single walker present the vertex labels are rewritten as Fock states, each showing the number of walkers on all three vertices, but the structure of the graph is unchanged. (c) Adding a second walker increases the number of vertices and results in a weighted graph with self-loops of weight g. (Unspecified edge weights are equal to 1.)

two vertices, and zero walkers on the third. As expected, the single-walker secondary graph is isomorphic to the primary graph, due to the correspondence between the singly occupied states $c_v^\dagger|\text{vac}\rangle$ and the vertex basis states $|v\rangle$. The number of vertices in the secondary graph corresponding to M walkers on an N-vertex primary graph is equal to the number of ways to arrange M identical objects into N groups. This task can be thought of as placing the M objects in a line and interspersing $N-1$ boundaries among them to create the N groups; this is sometimes referred to as the 'stars and bars' method because, for example, two of the combinations with $M=5$ and $N=3$ can be expressed as '$\star\star|\star|\star\star$' and '$|\star\star\star\star|\star$' (corresponding to the Fock states $|212\rangle$ and $|041\rangle$, respectively). In each string there are $M+N-1$ objects (stars and bars) leading to $(M+N-1)!$ arrangements, but since the M stars are identical, as are the $N-1$ bars, the total number of arrangements is reduced by factors of $M!$ and $(N-1)!$. Thus in general for the case of indistinguishable bosons, the M-walker secondary graph corresponding to

a primary graph on N vertices contains

$$N^{(M)} \doteq \frac{(M+N-1)!}{M!(N-1)!} = \binom{M+N-1}{M} \tag{6.4}$$

vertices. If the number of walkers scales with the size of the graph, say $M = kN$ for some integer $k > 0$, then neglecting the -1 term in Equation (6.4) and using Stirling's approximation indicates that the size of the secondary graph grows exponentially in the size of the primary graph:

$$\begin{aligned} N^{(M)} &\approx \frac{(M+N)!}{M!N!} = \frac{[(k+1)N]!}{(kN)!N!} \\ &\approx \frac{(k+1)^{N(k+1)}}{k^{Nk}} = \left[k\left(\frac{k+1}{k}\right)^{k+1}\right]^N, \end{aligned} \tag{6.5}$$

where the parenthetic base raised to the nth power is a monotonically increasing function of k that is strictly greater than unity. Numerically this base is found to equal 2 near $k \approx 0.3$. Thus by choosing $k \gtrapprox 1/3$, the Hilbert space dimension grows more quickly than required to encode $n = M$ qubits. This fact can be exploited to construct an encoding of 2^N computational states in the states of a graph with $O(N)$ vertices at half-filling, i.e. $k = 1/2$, providing an exponential savings in the number of vertices of the previously discussed schemes that require $\Omega(2^N)$ vertices.

6.1.2 Walker positions as computational states

Quantum walk-based schemes for quantum computation generally employ a position-based encoding, as seen in the schemes presented in Chapters 3 and 5. This results in a spatially delocalized qubit under circumstances where the quantum walk formalism is directly mapped to the circuit model. With multiple walkers the situation is no different, as the only degree of freedom available to each walker is its position on the primary graph.

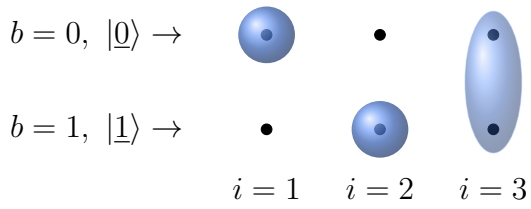


Figure 6.2: Cartoon example of a three-qubit computational state of the form $|\Psi\rangle \sim |\underline{0}\rangle \otimes |\underline{1}\rangle \otimes |\underline{+}\rangle$ encoded by three walkers on six vertices. The physical Hilbert space for such a system is 56-dimensional; eight of these dimensions encode computational basis states, and are coupled to a total of 24 physical basis states by the Hamiltonians prescribed to implement universal computation. The six vertices on which primary graphs are constructed of $|v_{i,b}\rangle$ for $i \in \{1, 2, 3\}$ and $b \in \{\underline{0}, \underline{1}\}$.

However, if there is a one-to-one mapping between walkers and qubits, then each walker requires access to only two vertices in order to encode the state of its qubit. That is, to specify an arbitrary n-qubit state under such a one-walker-per-qubit encoding, $M = n$ walkers require $N = 2M$ vertices. A cartoon example depicting this manner of encoding can be found in Figure 6.2 for $n = 3$ qubits.

In the scheme we propose in Reference [84], the primary graph includes a unique pair of vertices for each walker, of which there is one for each qubit to be simulated, corresponding directly to the computational states $|\underline{0}\rangle$ and $|\underline{1}\rangle$ of the encoded qubit. Gates are implemented by making instantaneous changes to the graph supporting the walkers, including those that explicitly couple to doubly occupied states in which two walkers interact on a single vertex, despite the fact that such states do not encode qubits. The graph is fixed except at the instants of change, so continuous dynamical control is not required. This results in discrete steps of continuous evolution on constant graphs, just as in the discontinuous single-walker proposal presented in Chapter 5. The main idea is to design a discrete sequence of graphs, such that at the beginning and end of evolution under the graphs, the spatial separation of the walkers is maintained with one walker per pair of vertices, yet during their evolution the walkers have the opportunity

to interact whenever two walkers occupy a single vertex. This allows states with doubly occupied vertices to be harnessed as an advantage in constructing logical gates, rather than having to be strictly avoided as a source of decoherence during gate operations as in previous proposals [91]. One way to envisage the system is as a two-dimensional optical lattice at half-filling that is sufficiently deep as to effectively eliminate tunnelling between sites, except when the potential is modified locally in order to add edges or self-loops to the primary graph.

Explicitly, the encoding of computational states in walker positions is defined as follows. Label the $2n$ vertices of the primary graph as $v_{i,b}$ with i indexing the qubit from $\{1,\ldots,n\}$, and b the computational bit values $\{\underline{0},\underline{1}\}$ The vacuum state of the graph is then written

$$|\mathrm{vac}\rangle = |0_{1,\underline{0}}0_{1,\underline{1}}\cdots 0_{n,\underline{0}}0_{n,\underline{1}}\rangle, \tag{6.6}$$

which can also be expressed using the convenient notation

$$|\mathrm{vac}\rangle = |00\rangle_1\cdots|00\rangle_n, \tag{6.7}$$

wherein the first and second entry of each ket is understood to correspond to the $b=\underline{0}$ and $b=\underline{1}$ states, respectively, for the qubit identified by the subscript. A walker is created on vertex $v_{i,b}$ by the operator $c^\dagger_{i,b}$.

According to Equations (6.4) and (6.5) with $k=1/2$, the dimension of the Hilbert space $\mathscr{H}$ for this system of $M=n$ walkers on $N=2n$ vertices is

$$N^{(M)} = \binom{3n-1}{n} \approx \left(\frac{3\sqrt{3}}{2}\right)^n \approx 2.6^n, \tag{6.8}$$

larger than the 2^n required computational basis states. The encoding of the computational basis is accomplished with a subset $\mathscr{H}_C \subset \mathscr{H}$, such that $|\mathscr{H}_C| = 2^n$. However,

the remainder of the space, $\mathscr{H}_\perp \doteq \mathscr{H} \setminus \mathscr{H}_C$, cannot be ignored; to generate entanglement between the encoded qubits, edges are added to the primary graph that result in connections on the secondary graph between vertices that encode computational basis states and vertices that do not. Since a physical state $|\Psi\rangle \in \mathscr{H}_C$ encodes a computational state but any state $|\alpha\rangle \in \mathscr{H}$ with non-zero support in $\mathscr{H}_\perp$ does not, the resulting evolution appears non-unitary from the computational point of view. However, the secondary graphs are constructed to exhibit periodicity or perfect state transfer such that at well-defined times during the evolution of the walkers, all probability amplitude on the graph returns to the computational space $\mathscr{H}_C$. That is, a set of walkers encoding a computational state evolve through states with no such encoding, yet at a later time once again encode a computational state, related to the first by the action of a computational unitary operator.

The ability to construct a graph Hamiltonian under this encoding that results in computational entanglement generation relies on the indistinguishability of the bosons involved as the operations of interchanging two walkers and of swapping them twice, returning them to their initial configuration, must be identical. This is crucial for preserving the mapping from the physical system to the computational space that requires one bosonic walker to be localized to two vertices of the primary graph.

Unless a computational operation is explicitly being performed, it is assumed that there are no edges in the primary graph, i.e. its edge set is empty, $E = \varnothing$. This default configuration is described by the system Hamiltonian

$$\mathcal{H}_0 \doteq \frac{g}{2} \sum_{i=1}^{n} \sum_{b=\underline{0}}^{\underline{1}} \mathcal{N}_{i,b} \left(\mathcal{N}_{i,b} - 1 \right), \tag{6.9}$$

containing only the always-on on-site inter-particle interactions. An n-walker physical

state encodes an n-qubit computational state if and only if

$$\sum_{b=\underline{0}}^{\underline{1}} \left| \langle \Psi | c_{i,b}^{\dagger} c_{i,b} | \Psi \rangle \right|^2 = 1 \tag{6.10}$$

for each i from 1 to n. That is, if there is exactly one walker per vertex pair $\{v_{i,\underline{0}}, v_{i,\underline{1}}\}$. Physical states satisfying this criterion are mapped onto computational ones in a canonical way:

$$|\underline{0}\rangle_i \leftrightarrow c_{i,\underline{0}}^{\dagger} |00\rangle_i = |10\rangle_i, \tag{6.11a}$$

$$|\underline{1}\rangle_i \leftrightarrow c_{i,\underline{1}}^{\dagger} |00\rangle_i = |01\rangle_i. \tag{6.11b}$$

The initial state of the system is taken to be

$$|\Psi_0\rangle \doteq \bigotimes_{i=1}^{n} c_{i,\underline{0}}^{\dagger} |\mathrm{vac}\rangle \leftrightarrow |\underline{0}\rangle^{\otimes n}, \tag{6.12}$$

a zero-energy eigenstate of $\mathcal{H}_0$ and therefore stationary when no edges are present in the primary graph.

6.2 Computing with multiple walkers

6.2.1 Single-qubit gates

Consider the ith of n encoded qubits, with computational basis states $|\underline{0}\rangle_i$ and $|\underline{1}\rangle_i$ encoded according to (6.11) in a physical state $|\Psi\rangle$ satisfying the single-particle condition of Equation (6.10). The single-qubit operations constructed in this section trivially preserve this condition, as they do not couple $|\Psi\rangle$ to states outside of the computational space.

An arbitrary X rotation can be implemented on qubit i by adding an edge of weight $\tau_{X,i}$ between the vertices $v_{i,\underline{0}}$ and $v_{i,\underline{1}}$. While this edge is present, the system Hamiltonian

is

$$\mathcal{H}_{X,i} = -\tau_{X,i}\left(c_{i,0}^{\dagger}c_{i,1} + \text{H.c.}\right) + \mathcal{H}_0. \tag{6.13}$$

The primary graph in this case contains $2(n-1)$ disconnected vertices and one copy of K_2, the connected two-vertex graph, on the vertices encoding qubit i. The action of $\mathcal{H}_{X,i}$ on the basis states (6.11) is simply

$$\mathcal{H}_{X,i} : |10\rangle_j \mapsto -\delta_{ij}\tau_{X,i}|01\rangle_j, \tag{6.14a}$$

$$|01\rangle_j \mapsto -\delta_{ij}\tau_{X,i}|10\rangle_j, \tag{6.14b}$$

where δ_{ij} is the Kronecker delta. In the computational space this acts as an X operator on qubit i,

$$\mathcal{H}_{X,i} : |\underline{0}\rangle_i \mapsto -\tau_{X,i}|\underline{1}\rangle_i, \tag{6.15a}$$

$$|\underline{1}\rangle_i \mapsto -\tau_{X,i}|\underline{0}\rangle_i, \tag{6.15b}$$

and has no effect on the other qubits encoded in $|\Psi\rangle$. The unitary operator generated by evolution under this Hamiltonian for a time t is $U_{X,i}(t) = R_{X,i}(-2\tau_{X,i}t)$, an X rotation of the ith qubit. Given sufficient control over the value of the edge weight $\tau_{X,i}$, an arbitrary rotation can be performed in a fixed time step. Conversely with even a single non-zero value of $\tau_{X,i}$ available, arbitrary angles of rotation can be achieved by enabling the edge for a suitable duration of $t = O(1/\tau_{X,i})$.

A Z rotation is created by the addition of a self-loop to a single vertex. This is equivalent to the addition of a local potential energy. Again consider qubit i, now with no edge between its vertices and instead a self-loop of weight $\mu_{Z,i}$ attached to vertex $v_{i,\underline{1}}$.

The physical Hamiltonian becomes

$$\mathcal{H}_{Z,i} = -\mu_{Z,i}\mathcal{N}_{i,1} + \mathcal{H}_0 \tag{6.16}$$

and corresponds to a primary graph of $2n$ disconnected vertices, with the attached self-loop on the second of the two vertices that encode qubit i. The Hamiltonian acts only on the $|\underline{1}\rangle_i$ computational basis state, according to

$$\mathcal{H}_{Z,i} : |\underline{1}\rangle_j \mapsto -\delta_{ij}\mu_{Z,i}|\underline{1}\rangle_j, \tag{6.17}$$

so the resulting unitary is $U_{Z,i}(t) = e^{i\mu_{Z,i}t/2}R_{Z,i}(\mu_{Z,i}t)$ — a Z rotation of qubit i, up to an unimportant overall phase. As with the X rotation, arbitrary angles of rotation can be achieved with control over either the self-loop weight $\mu_{Z,i}$, the time t, or a combination of the two.

To summarize, given an angle $\theta \in [0, 2\pi)$ the unitary $R_X(\theta)$ can be applied to qubit i by evolving under $\mathcal{H}_{X,i}$ for a time

$$t_{X,i}(\theta) \doteq \frac{4\pi - \theta}{2\tau_{X,i}}, \tag{6.18a}$$

and $e^{i\theta/2}R_Z(\theta)$ by evolving under $\mathcal{H}_{Z,i}$ for a time

$$t_{Z,i}(\theta) \doteq \frac{\theta}{\mu_{Z,i}}. \tag{6.18b}$$

Given sufficient freedom in the ability to set the tunnelling rates and on-site potentials, it is possible to enact either of these single-qubit gates on each qubit simultaneously, with different values of θ on each one, and have them finish at the same time. That is, given a set of angles θ_i and a choice of gates $U_i \in \{R_X, R_Z, I\}$, the values of $\tau_{X,i}$ and

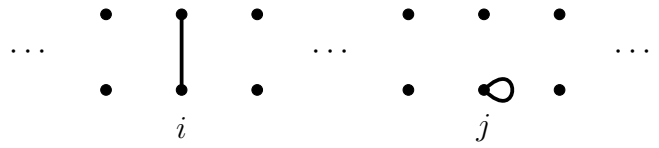


Figure 6.3: An n-walker graph on $2n$ vertices, on which appropriately initialized bosonic walkers will undergo X and Z rotations on the encoded qubits i and j, respectively.

$\mu_{Z,i}$ can be chosen for each qubit such that in a fixed time t, the operations $U_i(\theta_i)$ are simultaneously applied across all qubits $i \in \{1, \ldots, n\}$. Figure 6.3 contains an example multi-walker primary graph that applies an X rotation to qubit i and a Z rotation to qubit j. The combination of these operations allows for the execution of arbitrary single-qubit unitaries in three steps by decomposing the corresponding rotations of the Bloch sphere using Euler angles.

Note also that while these single-qubit operators are sufficient to implement a Hadamard operation

$$H \doteq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{6.19}$$

on qubit i in three steps, as

$$H_i = R_{X,i}(\pi/2) R_{Z,i}(\pi/2) R_{X,i}(\pi/2), \tag{6.20}$$

it is possible to obtain a Hadamard in a single step with the Hamiltonian

$$\mathcal{H}_{H,i} = -\mu_{H,i} \mathcal{N}_{i,0} - \tau_{H,i} \left(c_{i,0}^\dagger c_{i,1} + \text{H.c.} \right) + \mathcal{H}_0. \tag{6.21}$$

This simple approach of effectively turning on the Hamiltonians for X and Z rotations simultaneously results in the application of a Hadamard gate on qubit i, up to an overall

phase, at time

$$t_{H,i} = \frac{\pi}{2\sqrt{2}\tau_{H,i}} \tag{6.22}$$

if the applied-potential–to–hopping ratio is tuned to be $\mu_{H,i}/\tau_{H,i} = 2$. Not only does this one-step process require fewer dynamical controls, but for equal hopping and interactions terms (i.e. taking $J_{H,i} = J_{X,i}$ and $\mu_{Z,i} = \mu_{H,i}$) results in a run-time that is an order of magnitude shorter. This example is unlikely to be the only such shortcut to additional gates available by judicious choices of further Hamiltonians.

6.2.2 Generating entanglement

Computational entanglement between adjacent qubits can be generated through a walker interaction that leads to the simulation of a CPHASE(φ) gate, for φ taken from a specific set of values to be discussed. Since the gate acts on two qubits, four vertices of the primary graph must be considered, two for each of qubits i and j that are to be acted upon. An initial input for the gate is assumed to satisfy the condition of Equation (6.10) and therefore has exactly two walkers present on these four vertices. The physical Hilbert space in question, corresponding to two indistinguishable bosonic walkers on four sites, is 10 dimensional. Four basis states correspond to the computational basis, and these will be coupled to an additional four physical states by the graph implementing the CPHASE gate. The remaining two physical basis states can be ignored so long as the initial state is a computational one. It is notationally convenient at this point to drop the subscripts and write the Fock states on the four vertices in question as a single ket when there is no ambiguity from doing so, taking for example $|0110\rangle = |01\rangle_i|10\rangle_j$. The two-qubit

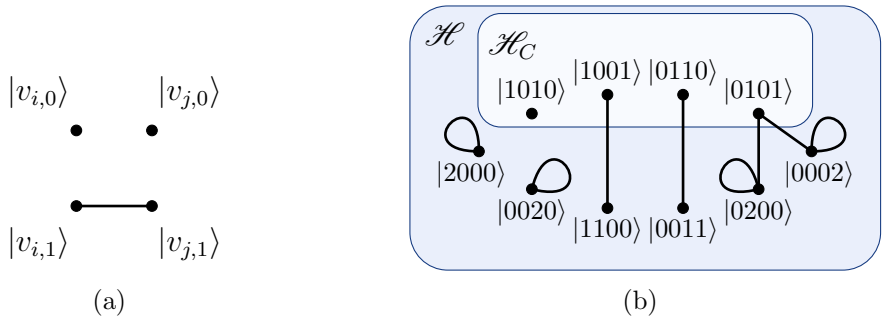


Figure 6.4: Primary and two-walker secondary graphs to generate entanglement. (a) The primary subgraph on which two walkers encoding qubits i and $i+1$ undergo a CPHASE operation has four vertices and a single edge. (b) The corresponding secondary graph has 10 vertices to represent the 10-dimensional Fock space of two bosons on four sites. Four of these encode the two-qubit computational basis and are coupled to an additional four. Self-loops arise due to the on-site interaction experienced by doubly occupied states.

computational space is encoded in the Fock number states

$$c_{i,0}^\dagger c_{j,0}^\dagger |0000\rangle = |1010\rangle \leftrightarrow |\underline{00}\rangle, \tag{6.23a}$$

$$c_{i,0}^\dagger c_{j,1}^\dagger |0000\rangle = |1001\rangle \leftrightarrow |\underline{01}\rangle, \tag{6.23b}$$

$$c_{i,1}^\dagger c_{j,0}^\dagger |0000\rangle = |0110\rangle \leftrightarrow |\underline{10}\rangle, \tag{6.23c}$$

$$c_{i,1}^\dagger c_{j,1}^\dagger |0000\rangle = |0101\rangle \leftrightarrow |\underline{11}\rangle. \tag{6.23d}$$

The six physical basis states $|1100\rangle$, $|0011\rangle$, $|2000\rangle$, $|0200\rangle$, $|0020\rangle$, and $|0002\rangle$ have no computational interpretation, as they each represent a state in which two walkers occupy the vertices assigned to encode a single qubit.

To implement the CPHASE gate, a single edge is placed between the two vertices encoding the $|\underline{1}\rangle$ states of qubits i and j, resulting in the primary graph shown in Figure 6.4(a).

The Hamiltonian for the multi-walker system becomes

$$\mathcal{H}_{\mathrm{CP},ij} = -\tau_{\mathrm{CP}} \left(c_{i,1}^{\dagger} c_{j,1} + \mathrm{H.c.} \right) + \mathcal{H}_0. \tag{6.24}$$

The action of this Hamiltonian on the physical basis states that encode computational ones is

$$\begin{aligned}
\mathcal{H}_{\mathrm{CP}} : |1010\rangle &\mapsto 0, && \text{(6.25a)} \\
|1001\rangle &\mapsto -\tau_{\mathrm{CP}} |1100\rangle, && \text{(6.25b)} \\
|0110\rangle &\mapsto -\tau_{\mathrm{CP}} |0011\rangle, && \text{(6.25c)} \\
|0101\rangle &\mapsto -\sqrt{2}\tau_{\mathrm{CP}} (|0200\rangle + |0002\rangle), && \text{(6.25d)}
\end{aligned}$$

where the 'i' and 'j' subscripts have been dropped for the present discussion, which is limited to these two qubits. The action of $\mathcal{H}_{\mathrm{CP}}$ on the remaining six physical basis states is

$$\begin{aligned}
|1100\rangle &\mapsto -\tau_{\mathrm{CP}} |1001\rangle, && \text{(6.25e)} \\
|0011\rangle &\mapsto -\tau_{\mathrm{CP}} |0110\rangle, && \text{(6.25f)} \\
|0200\rangle &\mapsto -\sqrt{2}\tau_{\mathrm{CP}} |0101\rangle + g|0200\rangle, && \text{(6.25g)} \\
|0002\rangle &\mapsto -\sqrt{2}\tau_{\mathrm{CP}} |0101\rangle + g|0002\rangle, && \text{(6.25h)} \\
|2000\rangle &\mapsto g|2000\rangle, && \text{(6.25i)} \\
|0020\rangle &\mapsto g|0020\rangle, && \text{(6.25j)}
\end{aligned}$$

leading to the secondary graph shown in Figure 6.4(b). Let $U_{\mathrm{CP}}(t) \doteq \exp(-\mathrm{i}\mathcal{H}_{\mathrm{CP}} t)$ be the time-evolution operator generated by $\mathcal{H}_{\mathrm{CP}}$. For any time t, the $|\underline{00}\rangle$ computational state evolves as $U_{\mathrm{CP}}(t)|1010\rangle = |1010\rangle$. This lack of any coupling to other basis states can be

seen in the secondary graph, shown in Figure 6.4(b), wherein the vertex corresponding to $|1010\rangle$ has degree zero. The state encoding $|\underline{01}\rangle$ couples outside of the computational basis to $|1100\rangle$, manifesting as a weighted copy of K_2 in the secondary graph. At time t,

$$U_{\text{CP}}(t)|1001\rangle = \cos(\tau_{\text{CP}}t)|1001\rangle + \mathrm{i}\sin(\tau_{\text{CP}}t)|1100\rangle \tag{6.26}$$

[cf. Equation (2.31)]. For this graph to simulate a computational operation, there can be no probability to find the walkers in a state that violates Equation (6.10), which translates to the secondary graph as a requirement that at the end of its evolution under $\mathcal{H}_{\text{CP}}$ a single walker initially confined to the vertices of $\mathscr{H}_C$ once again has no support on vertices outside this space. To accomplish this in light of Equation (6.26), the evolution time must be $t_{\text{CP},k} \doteq k\pi/\tau_{\text{CP}}$, $0 < k \in \mathbb{Z}$. This also satisfies the requirement that U_{CP} map $|\underline{10}\rangle$ to the computational basis. Specifically, for $|\psi\rangle \in \{|\underline{01}\rangle, |\underline{10}\rangle\}$ the resulting computational operation is $U_{\text{CP}}(t_{\text{CP},k})|\psi\rangle = (-1)^k|\psi\rangle$.

The evolution of $|0101\rangle$, the remaining physical basis state that encodes a computational one, is more complicated but treated in an identical manner. Constraints are placed on the parameters of the unitary evolution operator such that at time $t_{\text{CP},k}$ the overlap of the states $|0101\rangle$ and $U_{\text{CP}}(t_{\text{CP},k})|0101\rangle$ has unit magnitude, which is achieved by guaranteeing that the overlaps of $|0200\rangle$ and $|0002\rangle$ with $U_{\text{CP}}(t_{\text{CP},k})|0101\rangle$ vanish simultaneously. In fact, due to the symmetry of the subgraph induced by $|0101\rangle$, $|0200\rangle$, and $|0002\rangle$, these two overlaps are identical at all times:

$$\langle 0200|U_{\text{CP}}(t)|0101\rangle = \langle 0002|U_{\text{CP}}(t)|0101\rangle = \mathrm{i}\sqrt{\frac{8\tau_{\text{CP}}^2}{16\tau_{\text{CP}}^2 + g^2}}\mathrm{e}^{-\mathrm{i}gt/2}\sin\left(\frac{t}{2}\sqrt{16\tau_{\text{CP}}^2 + g^2}\right). \tag{6.27}$$

For a non-zero edge weight τ_{CP} and finite particle interaction strength g, this quantity

vanishes at $t = t_{\text{CP},k} = k\pi/\tau_{\text{CP}}$ if and only if

$$\frac{k\pi}{2\tau_{\text{CP}}}\sqrt{16\tau_{\text{CP}}^2 + g^2} \stackrel{!}{=} l\pi, \tag{6.28}$$

for some integer $l > 0$. Equation (6.28) is satisfied whenever the ratio of the on-site interaction strength g to the hopping rate τ_{CP} is

$$\frac{g}{\tau_{\text{CP}}} \stackrel{!}{=} 2\sqrt{\frac{l^2}{k^2} - 4} \tag{6.29}$$

where it must be the case that $l \geq 2k$ (with equality only when there are no inter-walker interactions, $g = 0$).

Given integers k and l satisfying these conditions, the evolution of the computational state $|\underline{11}\rangle$ is determined by

$$\langle 0101|U_{\text{CP}}|0101\rangle = \exp\left[-\mathrm{i}\pi\left(l + \sqrt{l^2 - 4k^2}\right)\right]. \tag{6.30}$$

Thus the computational unitary U_{CP} can be written as

$$U_{\text{CP}} = \text{diag}\left[1, (-1)^k, (-1)^k, \mathrm{e}^{-\mathrm{i}\pi(l+\sqrt{l^2-4k^2})}\right]. \tag{6.31}$$

Whenever the factor $l + \sqrt{l^2 - 4k^2}$ is not an integer, the result is an entangling gate equal to

$$U_{\text{CP}} = \begin{cases} \textsc{cphase}(\varphi_{k,l}), & k \text{ even}; \\ (Z \otimes Z)\textsc{cphase}(\varphi_{k,l}), & k \text{ odd}; \end{cases} \tag{6.32}$$

where

$$\varphi_{k,l} \doteq -\pi\left(l + \sqrt{l^2 - 4k^2}\right). \tag{6.33}$$

Note that an entangling gate would also arise if both k and the factor $l + \sqrt{l^2 - 4k^2}$ were

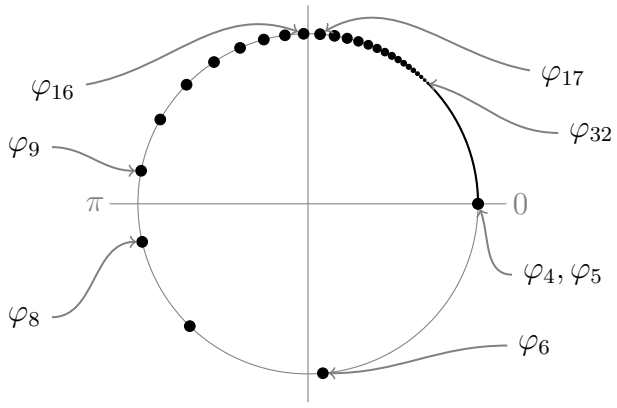


Figure 6.5: Distribution of phase angles φ_l satisfying Equation (6.34). Except for the two trivial phases φ_4 and φ_5, no φ_l is a rational multiple of π. For large l the phase angle goes as $-8\pi/l \pmod{2\pi}$, returning to zero as $l \to \infty$, which corresponds to the limit of zero hopping.

odd integers, however it can be shown that this is impossible. To minimize the time required while simulating a standard CPHASE gate, let $t_{\text{CP}} \doteq 2\pi/\tau_{\text{CP}}$. In this case the available controlled-phase angles are

$$\varphi_l \doteq -\pi \left(l + \sqrt{l^2 - 16} \right), \tag{6.34}$$

which can be seen as points on the unit circle in Figure 6.5. The angles φ_4 and φ_5 are congruent to zero, modulo 2π, and thus implement only local unitaries. As l tends to infinity, corresponding to the limit of zero hopping, the resulting phases tend to zero, but for all $l > 5$ the result is an entangling gate. Of particular interest are the phases $\{\varphi_9, \ldots, \varphi_{16}\}$ that fall between $\pi/2$ and π; according to Theorem 1.3 they can be used to simulate an arbitrary two-qubit gates with at most six applications of U_{CP}. Any one of these gates in combination with the arbitrary single-qubit operations of Section 6.2.1 provides an instruction set for universal two-qubit computation under the Bose–Hubbard model.

6.2.3 Adding a SWAP gate

The set V of $2n$ vertices on which the primary graphs discussed so far are defined has no intrinsic spatial distribution, and as such the edge used to generate entanglement between qubits i and j can simply be considered to be added regardless of the values of i and j. However, given a specific arrangement of the vertices, such as might be provided by an optical lattice, it may be possible to only entangle neighbouring pairs of qubits. In this case, a SWAP gate is required for universal computation in addition to the gate set already described.

A variation of the X rotation performed by $\mathcal{H}_{X,i}$ allows for a straightforward implementation of SWAP by the simultaneous addition of edges between the vertices encoding $|\underline{b}\rangle_i$ and $|\underline{b}\rangle_{i+1}$, for each $b \in \{0,1\}$. The system Hamiltonian to swap qubit i with its neighbour $i+1$ is

$$\mathcal{H}_{S,i} = -\tau_S \left(c_{i,0}^\dagger c_{i+1,0} + c_{i,1}^\dagger c_{i+1,1} + \text{H.c.} \right) + \mathcal{H}_0. \tag{6.35}$$

This acts non-trivially on the four two-walker basis states satisfying (6.10), coupling the computational space to all six of the remaining physical basis states. As with the analysis of the CPHASE gate, conditions can be placed on the available parameters to guarantee that the action of the operation restricted to the computational space is unitary at the end of the evolution. The action of $\mathcal{H}_S$ on $|1001\rangle$ and $|0110\rangle$ requires that the SWAP time be set to

$$t_{S,k} = \frac{(2k+1)\pi}{2\tau_S}, \quad 0 \leq k \in \mathbb{Z}. \tag{6.36a}$$

Under this restriction, the action of $\mathcal{H}_S$ on $|1010\rangle$ and $|0101\rangle$ further requires that

$$\frac{g}{\tau_S} = 4\sqrt{\frac{l^2}{(2k+1)^2} - 1}, \quad 2k+1 < l \in \mathbb{Z}. \tag{6.36b}$$

When these conditions are satisfied, the action of $U_S(t) \doteq \exp(-\mathrm{i}\mathcal{H}_S t)$ at time $t = t_{S,k}$ block diagonalizes such that its effect on the computational space is that of

$$U_S(t_{S,k}) = \begin{pmatrix} e^{-\mathrm{i}\alpha\pi} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & e^{-\mathrm{i}\alpha\pi} \end{pmatrix} \tag{6.37}$$

with

$$\alpha \doteq l + \sqrt{l^2 - (2k+1)^2}. \tag{6.38}$$

In general this provides a second entangling gate, unless α is an integer. In that case, if α is even then (6.37) is equivalent to $(Z \otimes Z)$SWAP, and if α is odd then the resulting gate is $-$SWAP.

The problem of finding integral values for k and l that are consistent with Equations (6.36) while resulting in an integral value of α reduces to finding Pythagorean triples, of which there are infinitely many. One such combination, which results in the minimal time for the operation, is $k = 1$ and $l = 5$. This pair leads to the parameters $t_S = 3\pi/(2\tau_S)$ and $g/\tau_S = 16/3$, and implements a two-qubit SWAP gate, up to an overall phase.

6.2.4 Multiple discontinuous walkers

Implementing a sequence of computational gates under this Bose–Hubbard-based model requires that a set of potentials be toggled on and off in a prescribed order, affecting the evolution of a set of quantum walkers in the process. This is in the same spirit as the single-walker discontinuous walk scheme proposed in Reference [77] and the previous chapter. However, it only requires linear growth of the number of vertices in the primary

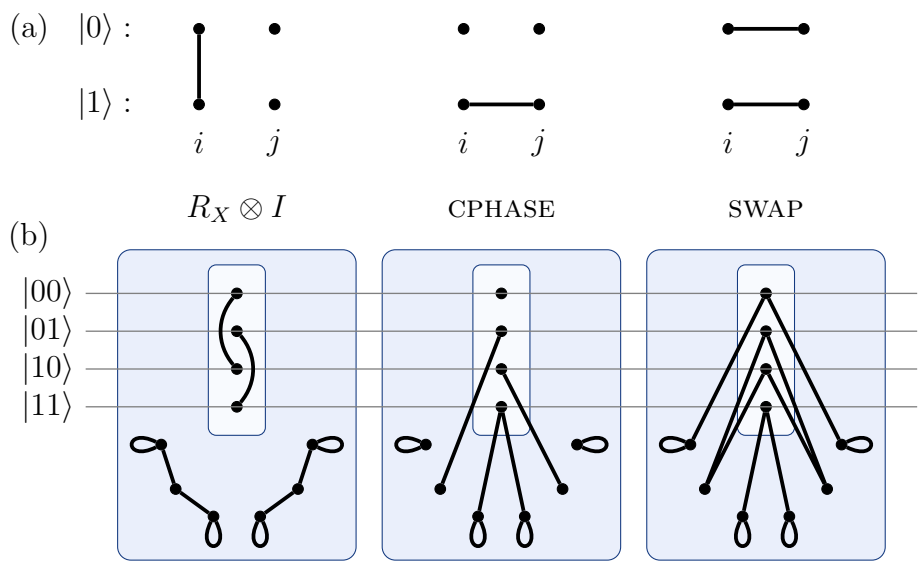


Figure 6.6: Sequence of multi-walker primary graphs viewed as a discontinuous walk. (a) Three primary graphs on two qubits that can be enabled in turn. (b) The corresponding secondary graphs in each case, with vertices that encode computational states identified with rails in analogy to the single-walker discontinuous scheme.

graph supporting the walkers, as a function of the number of qubits to be simulated. Figure 6.6 demonstrates how a time sequence of two-walker primary graphs on four vertices corresponds to a single discontinuous walker on a larger secondary graph, drawing a connection to the rail model used by all of the single-walker schemes for universal computation. In this case the number of rails simulated by the secondary graphs is equal to the number of vertices on the primary one since $2n = 2^n$ for $n = 2$. Figure 6.7 shows what a single step of a discontinuous walk on three qubits looks like, on six vertices in the primary multi-walker case and on eight rails passing through the vertices circled in lighter grey in a 56-dimensional space for a single walker on the resulting secondary graph.

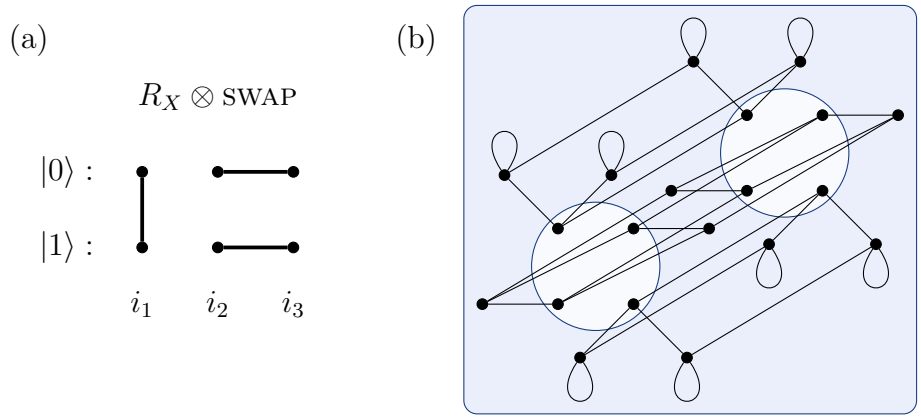


Figure 6.7: Comparison of the primary graph and three-walker secondary graph for a computation on three qubits. (a) On the primary graph, three edges among six vertices are sufficient to implement the unitary $R_X \otimes \text{SWAP}$. (b) The resulting secondary graph has 56 vertices, 20 of which are connected to the eight encoding computational basis states and shown here.

6.3 Measurements and error correction

In contrast to single-walker computational schemes, the Bose–Hubbard-based multi-walker model presented here allows for the measurement of p qubits with localized measurement operators. To extend the single-walker measurement operators of Equation (5.24) to the many-walker case, define

$$P_{i,b}^{(k)} \doteq (c_{i,b}^\dagger)^k |0_{i,b}\rangle\langle 0_{i,b}| (c_{i,b})^k = |k_{i,b}\rangle\langle k_{i,b}|, \tag{6.39}$$

the projector onto the state with exactly k walkers on vertex $|v_{i,b}\rangle$. The qubit encoded on vertices $|v_{i,0}\rangle$ and $|v_{i,1}\rangle$ can then be measured with the operators

$$M_{i,\underline{0}} \doteq P_{i,\underline{0}}^{(1)} \otimes P_{i,\underline{1}}^{(0)} \otimes I_{\bar{i}}, \tag{6.40a}$$

$$M_{i,\underline{1}} \doteq P_{i,\underline{0}}^{(0)} \otimes P_{i,\underline{1}}^{(1)} \otimes I_{\bar{i}}, \tag{6.40b}$$

$$M_{i,\text{err}} \doteq I - M_{i,\underline{0}} - M_{i,\underline{1}}, \tag{6.40c}$$

where $I_{\bar{i}}$ is the identity operator on every vertex not labelled by i. If the physical state $|\Psi\rangle$ of the system encodes a computational state as it is assumed to, then there is exactly one walker on the vertices encoding qubit i. That is,

$$|\langle\Psi|M_{i,\underline{0}}|\Psi\rangle|^2 + |\langle\Psi|M_{i,\underline{1}}|\Psi\rangle|^2 = 1, \tag{6.41}$$

so a projective measurement places qubit i in either the $|\underline{0}\rangle$ or $|\underline{1}\rangle$ state with certainty. The remainder of the qubits are unaffected. The possible role of $M_{i,\mathrm{err}}$ is discussed at the end of the section.

With the ability to perform single-qubit measurements on the multi-walker system in a straightforward manner comes the ability to implement quantum error-correcting codes using ancillary qubits to perform syndrome measurements, as discussed in Section 1.5. A straightforward adaptation of the qubit-encoding scheme provides a natural implementation of such codes, discussed here for the seven-qubit Steane code in particular, when the vertices of the primary graph are arrayed in a three-dimensional lattice.

Consider extending the $2 \times n$ grid employed thus far into the third dimension, and creating a $4 \times 7 \times n$ array of vertices as depicted in Figure 6.8. Each pair of vertices encoding a single qubit in the original version of the scheme, as in Region A of Figure 6.8, has an additional six pairs extending from it along the third dimension of the lattice. The resulting total of seven pairs at a given position i within the lattice, Region B, allows for the creation of a single logical qubit. Six further ancillary qubits, Region C, can be entangled with the physical qubits of the logical one in order to implement syndrome measurements. Making the vertices that correspond to the $|\underline{1}\rangle$ states in the two regions adjacent allows entanglement to be generated between them by the CPHASE gate with the straightforward addition of edges between the logical region and the ancillary one. With respect to the circuit in Figure 1.4 for the seven-qubit Steane code, Region A corresponds

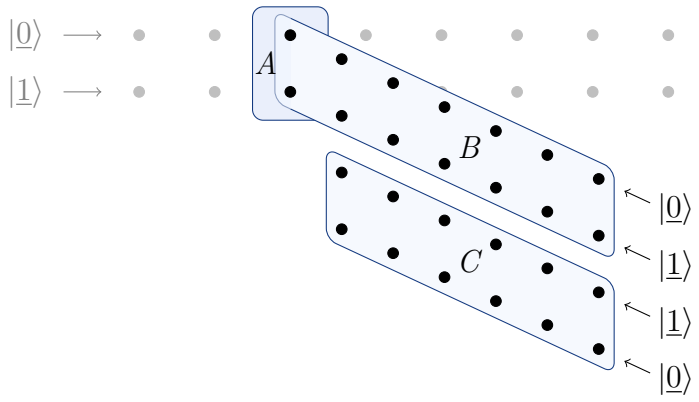


Figure 6.8: Addition of the seven-qubit Steane quantum error-correcting code to the Bose–Hubbard scheme for quantum computation. The scheme described in Section 6.2 takes place on the horizontal rows of light grey vertices, two of which have been highlighted in Region A. Seven walkers on the 14 vertices in Region B (including those also in A) provide seven physical qubits with which to encode a single logical qubit. Region C provides six further ancillary qubits that can be entangled with those of the logical qubit in order to perform syndrome measurements as a sequence of single-qubit measurements. Note that the ordering of the $|\underline{0}\rangle$ and $|\underline{1}\rangle$ rows has been reversed in C with respect to B; this allows a straightforward method of generating entanglement between the two regions by way of the CPHASE gate that results when an edge is placed between the $|\underline{1}\rangle$ vertices of neighbouring qubits.

to the rail containing the initial input state $|\psi\rangle$. Imagine folding the circuit in half along that rail, so that the six additional encoding qubits lie on top of (but displaced from) the six ancillary qubits used for syndrome measurements. This configuration is the motivation for the relative placements of the remainder of the encoding qubits in Region B and the ancillary ones in Region C.

Logical operations on these additional qubits in this new configuration can of course be performed in the same manner as on the original qubits described in Section 6.2. Those gates are dependent on certain couplings between pairs of vertices, but are not inherently reliant on a grid- or lattice-like structure in the placement of those vertices; such a setting merely provides a useful visualization method, and yields an obvious tie to possible extant physical systems, which is discussed in Section 6.4. They can therefore all be applied between pairs of qubits in this new arrangement as well. From the point of view of quantum walks on graphs, wherein edges can be placed between arbitrary pairs of vertices, this arrangement is purely for visual convenience, and a circuit akin to Figure 1.4 for the seven-qubit code can be implemented directly.

More physically motivated scenarios in which the vertices correspond to spatial locations are discussed in the following section. In such situations, if interactions can only be engineered between neighbouring sites of the lattice then one of two additions is required to allow the coupling of qubits in Region C to the appropriate subsets of those in Region B. One option could be to shift the portion of the lattice containing Region C laterally, aligning its sites with different locations in Region B at different times. If this is not feasible in a given setup, then the SWAP gate discussed in Section 6.2.3 can be employed instead.

Having seen that the multi-walker Bose–Hubbard-based approach to universal computation provides the advantage of being able to implement computational error correction, previously unaddressed within quantum walk-based quantum computation, it is also im-

portant to note that the dynamical nature of the underlying graph introduces a second potential source of error. If a gate is timed incorrectly, it is possible for the system to end up in a physical state that does not encode a computational one; in addition to the standard quantum errors of being flipped and dephased, a qubit can be lost altogether. When this possibility is allowed for, the third measurement operator $M_{i,\mathrm{err}}$ becomes important. When the single-qubit measurement is made, it becomes possible to obtain a result other than $|\underline{0}\rangle$ or $|\underline{1}\rangle$, and doing so indicates that an error has occurred. This provides a simple method for detecting the loss of a qubit, but not a mechanism for recovering from it. The possibility of losing the walker exists in the original proposal for universal quantum computation by a single quantum walker [62], and a similar timing issue is present in the discontinuous-walk scheme of Chapter 5 and Reference [77], as well as any potential physical scheme for the simulation of a discrete-time quantum walk with a continuously evolving physical system [73]. In photonic implementations of discrete-time quantum walks, timing of operations manifests instead in the spatial separation between optical elements [75]. These are not fundamental limitations on quantum walk-based computing, but must be considered in any serious attempt to engineer such systems.

6.4 Considerations for a physical implementation

One feature of a physical system that is to implement this proposal in a straightforward manner is the ability to provide a $2 \times n$ lattice — or three-dimensional lattice, if that method of quantum error correction is chosen — such that the tunnelling amplitude between neighbouring sites is close to zero when no gate is being enacted. This corresponds to the qubits' having a long coherence time. In practice this can be accomplished in either the 2D or 3D case by creating a three-dimensional lattice with isolated wells, and ignoring any vertices outside of the primary graph to be implemented.

The additional requirements of state preparation, manipulation, and read-out are common to any quantum computer, and are further discussed here in the context of this proposal. State preparation is accomplished by the initialization of the $|0\rangle^{\otimes n}$ state, by loading one boson onto the first of each pair of sites in the primary graph. To manipulate the encoded qubits, it must be possible to selectively increase the tunnelling amplitude between given adjacent sites in order to enact X, CPHASE, and SWAP gates, and to change on-site potentials to enact Z gates. Finally to read out the result of a computation it must be possible to measure the positions of the bosonic walkers within the lattice, as discussed in Section 6.3.

The experimental scheme proposed in Reference [91] includes many of the features required, though in sharp contrast to the current proposal it makes use of adiabatic processes for gate executions. A combination of this setup with the sudden potential-landscape changes discussed in Reference [92] is more appropriate to the Bose–Hubbard scheme, and there have been significant experimental advances in the intervening years that offer the promise of a proof-of-principle implementation. There is an obvious connection between this scheme and optical-lattice experiments, and several experiments satisfy the above requirements. One option may be to use a liquid-crystal display (LCD) as a spatial light modulator [93]. Such devices generate arrays of microtraps holographically based on the pattern of opacity and transparency present on the easily programmed LCD screen. The traps have been used to store single neutral atoms per site, address individual sites, and measure the locations of trapped atoms within the lattice, thus providing a means to create the primary graph as well as implement preparation, manipulation, and read-out.

Another possibility is to combine a set of recently demonstrated experimental capabilities. Wide lattice spacings on the order of 5 µm have been achieved [94], providing a long coherence time to the sites and effectively approximating the infinitely deep lattice

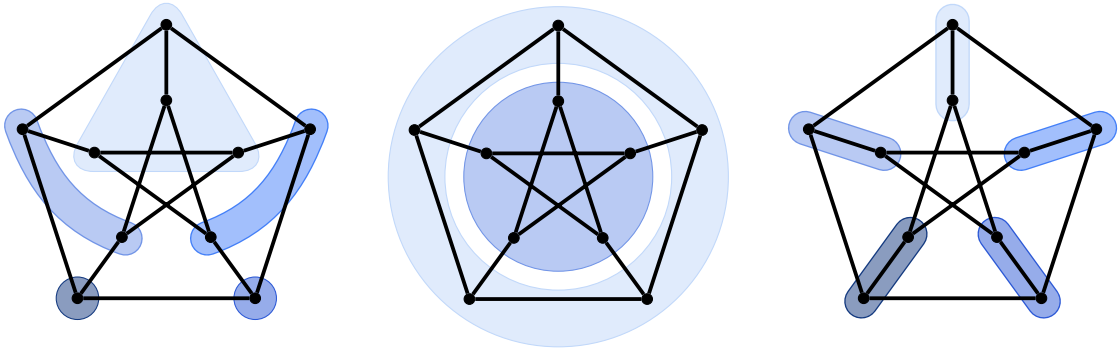
of the primary graph when no edges are present. A quantum gas microscope, employing a high-numerical-aperture lens, has been used to image individual sites in a traditional optical lattice [95]. Repurposing such a system to focus a laser to a similar resolution would provide a method of manipulating qubits by addressing single sites in the case of a Z gate, or modifying the potential between sites in the case of X and CPHASE gates. Most recently, arbitrary configurations of atom positions within a lattice have been implemented [96], which in particular would allow for the straightforward preparation of the initial $|0\rangle^{\otimes n}$ state as a single straight line, one atom wide. In each of these experiments, measurement of the positions of the trapped atoms is also performed, allowing the final result of the computation to be read out in each case.

Chapter 7

Graph-theoretic approaches to particles on lattices

As has been made clear in the previous chapters, the link between the behaviour of quantum walkers and the graphs on which they walk is significant; no study of the former can help but make reference to the latter. It is not surprising then that certain results from the realm of graph theory have been employed in the analysis of quantum walks. One such concept in particular, which forms the primary topic of this chapter, is the equitable partitioning of graphs, which has been used in the search for graphs that admit perfect state transfer [44, 97–99] as discussed in the next section. Until the present work, equitable partitioning has been limited primarily to the case of simple graphs, with some recent extension to the case of signed graphs, in which edge weights can be ± 1 [100]. Signed graphs arise in the context of multiple fermionic walkers on a graph; in the case of bosonic walkers more general edge weights arise, as for example in Figure 6.1(c). With this in mind, the purpose of the current chapter is to describe the expanded formalism I have developed for the equitable partitioning of graphs with arbitrary real self-loop and edge weights, positive or negative. These results appear in Section 7.2, following a review of partitions on simple graphs in Section 7.1.

A partition on a graph is a division of its vertices into disjoint subsets, referred to as 'cells'. Examples include the grouping of the vertices of a hypercube according to their distance from a given vertex, or into two sets that demonstrate the graph is bipartite [cf. Figures 2.3(a) and (b), respectively], as well as the grouping of vertices in the glued-trees graphs of Figure 3.1 by their distance from the input vertex. An equitable partition is one that groups vertices according to some symmetry of the graph, such that the number of edges joining a vertex in one cell to another is the same for all vertices in the first.



(a) (b) (c)

Figure 7.1: Example partitions on the Petersen graph. The partitions in (a) and (b) are equitable, while (c) is inequitable despite the apparent symmetry of the partition. This can be seen because, for example, the upper vertex of the upper cell is connected to the cells to its left and right, but the lower vertex of the upper cell is connected instead to the lowest two cells.

(This must also hold in the case where the first and second cells are in fact the same.) Figure 7.1 shows three partitions on a graph, only two of which are equitable. Given a graph and an equitable partitioning of its vertices, one can create a second 'collapsed' graph in which there is one vertex for each cell in the partition, and the edge weights between these collapsed vertices depend on the number of edges between their parent cells. The collapsed graph shares certain properties with the original, in particular with respect to their eigenvalue spectra and aspects of quantum walks upon them, as made explicit below.

In the context of multiple quantum walkers on a graph, equitable partitioning of the corresponding secondary graph is a useful tool with which to analyse the system. For example, the secondary graph resulting from multiple distinguishable walkers on a given primary graph can be collapsed to the secondary graph for the same number of indistinguishable walkers on the same primary graph. The procedure corresponds to the symmetrization of the many-body wave function in the standard transition from distinguishable- to indistinguishable-particle systems. Furthermore, it is shown below

that the largest eigenvalue of the adjacency matrix for bosonic walkers is preserved under collapse. That is, the largest eigenvalue of the collapsed graph is equal to that of the original. Since the system Hamiltonian is given by the negative of the adjacency matrix, the largest eigenvalue of the graph corresponds to the ground-state energy of the many-boson system. This same result appears to also hold for fermionic systems in my experience, but whether a general proof exists in this case remains an open question.

The properties of periodicity and perfect state transfer on a collapsed graph also yield information about evolution of a walker on the original. If there is perfect state transfer on the collapsed graph between two vertices corresponding to cells that each contain a single vertex, such as the two lowest cells in Figure 7.1(a), then there is perfect state transfer between those two vertices on the larger uncollapsed graph as well. When there is PST between two vertices of the collapsed graph that do not correspond to individual vertices of the original graph, one can still construct a superposition over the vertices of one cell that will evolve into a superposition with support only in the other.

7.1 Equitable partitioning of graphs

A *partition* $\Pi = \{C_i\}_{i=1}^{M}$ of a simple graph $G = (V, E)$ is a set of M disjoint subsets C_i of V, called the *cells* of the partition, such that the union of the C_i is V. Each vertex $v \in V$ is an element of exactly one cell, which can be indexed by $\tilde{v}$; that is, $\tilde{v}$ is defined to be the unique i such that $v \in C_i$. A cell containing exactly one vertex, C_i such that $|C_i| = 1$, is referred to as a *singleton* cell. A partition of a simple graph is *equitable* if the number of neighbours in a cell C_i of a vertex $v \in C_j$ is a constant b_{ij}, independent of the choice of v from C_j [32]. Equitable partitions of simple graphs have been studied extensively in the context of perfect quantum state transfer, in part because it is possible to use an equitable partition to 'collapse' the original N-vertex graph G to a smaller

M-vertex graph G/Π, in which each vertex represents all of the equivalent vertices of the original graph in one cell of the partition, such that certain key features, in particular unitary evolution between singleton cells, are preserved.

An example of an analysis performed with such a collapse is the so-called column method for reducing the glued trees of Figure 3.1 to a weighted line by replacing all vertices a fixed distance from one root by a single vertex [35, 56]. The same idea has also been applied to the 2^k-vertex hypercube in order to reduce it to a $(k+1)$-vertex line [41], though the formalism of equitable partitioning was not employed in either case. The relative ease with which the collapse of the hypercube can be performed without knowledge of equitable partitions makes it an excellent concrete example with which to introduce the formalism. Consider for example the cube shown in Figure 7.2 along with its collapse to a weighted line. To perform the collapse, one vertex is singled out and the eight vertices are partitioned into the four cells shown, such that vertices within a cell are all the same distance from the chosen one. This automatically creates two singleton cells, one for each of the chosen vertex and its antipode.

The *partition matrix* P corresponding to a given partition is the matrix that maps each N-dimensional basis vector of the vertex basis to an M-dimensional basis vector in the cell basis,

$$P \doteq \sum_{i=1}^{M} \sum_{v \in C_i} |v^{(N)}\rangle\langle i^{(M)}|, \tag{7.1}$$

where the parenthetic superscripts indicate the dimensions of the vectors. Of more use in the context of perfect state transfer and the evolution of quantum walks on graphs is the *normalized partition matrix* Q, constructed similarly according to the definition

$$Q \doteq \sum_{i=1}^{M} \frac{1}{\sqrt{|C_i|}} \sum_{v \in C_i} |v^{(N)}\rangle\langle i^{(M)}|, \tag{7.2}$$

Two useful facts about the normalized partition matrix are that $Q^T Q = I_M$, and that

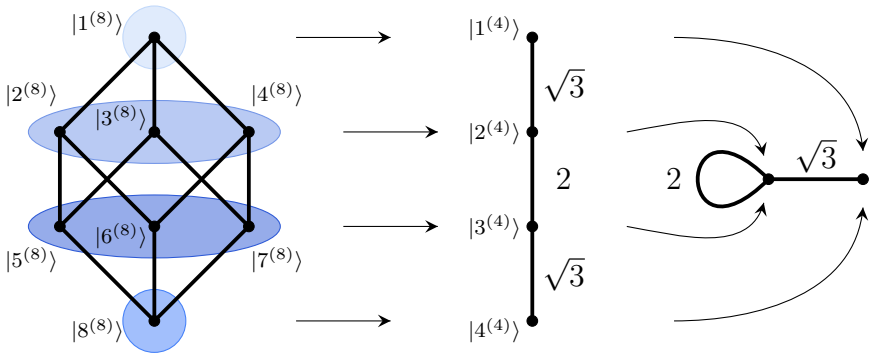


Figure 7.2: Collapsing the 3-cube to a path under equitable partitioning. With the upper-most vertex chosen to be put in a singleton cell, the remaining vertices are grouped with those the same distance from it. Each cell is replaced by a single vertex, and the edge weight between cells C_i and C_j is set to $\sqrt{d_{ij}d_{ji}}$, where d_{ij} is the number of neighbours in cell C_j of a vertex in C_i. The superscripts denote the dimensions of the vectors of the vertex basis. With a weighted equitable partition, the line can be collapsed further, by folding it in half.

$[QQ^T, A] = 0$ [45]. In general $QQ^T \neq I_N$, unless every cell of Π is a singleton so that $M = N$ and $Q = I_N$.

For the specific case in Figure 7.2, the normalized partition matrix is

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{7.3}$$

and it is straightforward to show that the adjacency matrices A_{cube} and A_{line} of the graphs

in Figure 7.2 are related by

$$A_{\text{line}} = Q^T A_{\text{cube}} Q. \tag{7.4}$$

Furthermore, it can be shown that the evolution between endpoints of the line is equivalent to that between antipodal vertices of the cube,

$$\langle 4^{(4)}|\mathrm{e}^{\mathrm{i}A_{\text{line}}t}|1^{(4)}\rangle = \langle 8^{(8)}|\mathrm{e}^{\mathrm{i}A_{\text{cube}}t}|1^{(8)}\rangle. \tag{7.5}$$

The equivalence between evolution on a graph G and on the quotient graph G/Π collapsed from G with respect to the equitable partition Π is a general feature of this procedure, and one of the primary motivations for the investigation of equitable partitioning. In particular, it is known [98] that if u and v are in singleton cells of Π then

$$\langle \tilde{u}|\mathrm{e}^{-iQ^T AQt}|\tilde{v}\rangle = \langle u|\mathrm{e}^{-iAt}|v\rangle. \tag{7.6}$$

Here the dimensional superscripts have been dropped as there is no ambiguity from doing so.

In the hypercube-to-line example the collapsed graph is exponentially smaller than the original graph, yet the evolution between two identified vertices of interest is identical. Analysis of one system can provide exact solutions to certain questions about an exponentially larger system. In this case the n-hypercube is the secondary graph corresponding to n non-interacting distinguishable walkers on P_2. This can be seen by making the identification that the binary labelling of the vertices from 0 to 2^{n-1} as in Figure 2.3 can also be interpreted as describing whether each particle is on the left or right vertex; for example the vertex $|101\rangle$ of the secondary graph corresponds to the state in which the first and third walkers are on the right-hand vertex, and the second walker is on the left. If instead the walkers are indistinguishable and bosonic then instead of

$N = 2^n$ basis states there are $n + 1$ states, corresponding to zero walkers on the right, one walker on the right, and so on up to n walkers on the right. These states and the allowed transitions among them encode the line on $n + 1$ vertices, with weights given by the Bose enhancement factors associated with a walker's being annihilated on one site and created on the other. In general, collapses of unweighted graphs result in weighted ones, as does the creation of secondary graphs for systems of multiple indistinguishable walkers, whether or not the primary graph is weighted. To further collapse the result of an equitable partitioning of an unweighted graph, or to collapse a multi-walker secondary graph, a new definition of an equitable partition is required that takes into account non-unit edge weights. Equipped with such a tool, certain properties of large graphs may be determined from the analysis of smaller—perhaps exponentially smaller—graphs.

7.2 Equitable partitions of weighted graphs

The primary goal of the current section is to extend the established framework of graph collapse and analysis by equitable partitioning to the case of weighted graphs by providing a generalization of the criterion for being equitable, and proving that certain theorems continue to hold in the broader context. Additionally a theorem is presented that guarantees that the largest eigenvalues of the collapsed and uncollapsed graphs are identical; since the Hamiltonian of a system evolving on the graph is given by the negative of the adjacency matrix, this means that the ground-state energies of the two systems are the same.

Let $G = (V, E, w)$ be a connected, weighted, undirected graph on N vertices with corresponding adjacency matrix $A(G)$, and let $\Pi = \{C_i\}_{i=1}^{M}$ a partition of V. The definition of an equitable partition for an unweighted graph makes use of the degree of each vertex; a similar concept can be found for the case of a weighted graph. Define the *weight* of a

vertex v in a weighted graph to be

$$\omega(v) \doteq \sqrt{\sum_{u=1}^{N} A_{uv}^2}, \tag{7.7}$$

which in the unweighted case reduces to $\sqrt{d_v}$, the square root of the degree of v. The normalized weight of a vertex with respect to its containing cell is given by

$$\Omega(v) \doteq \frac{\omega(v)}{\sqrt{\sum_{u \in C_{\tilde{v}}} \omega^2(u)}} \doteq \frac{\omega(v)}{\omega(C_{\tilde{v}})}, \tag{7.8}$$

where the weight of a cell has been defined as the norm of the M-vector of the weights of all vertices in the cell. The normalized partition matrix Q is then defined as

$$Q \doteq \sum_{v=1}^{N} \Omega(v) |v\rangle\langle\tilde{v}|. \tag{7.9}$$

Theorem 7.1 provides the main result that extends the notion of equitable partitioning to weighted graphs. Its proof makes use of the following lemma.

Lemma 7.1. *Let Q be the normalized partition matrix describing a partition $\Pi = \{C_i\}_{i=1}^{M}$ of a weighted undirected connected graph G on N vertices. Then $Q^T Q = I_M$, the $M \times M$ identity.*

Proof. By the definition of Q, Equation (7.9),

$$Q^T Q = \sum_{v=1}^{N} \sum_{w=1}^{N} \Omega(v)\Omega(w) |\tilde{v}\rangle\langle v\,|w\rangle\langle\tilde{w}| = \sum_{v=1}^{N} \Omega^2(v) |\tilde{v}\rangle\langle\tilde{v}|. \tag{7.10}$$

This is a sum of N different $M \times M$ matrices with $N \geq M$, so some of the matrices appear multiple times with different coefficients. Specifically, if two vertices u and v belong to the same cell then $\tilde{u} = \tilde{v}$, so it is natural to sum only over the cells,

instead of over all vertices. Doing so and inserting Equation (7.8), the definition of Ω, yields

$$Q^T Q = \sum_{i=1}^{M} \sum_{v \in C_i} \frac{\omega^2(v)}{\omega^2(C_{\tilde{v}})} |\tilde{v}\rangle\langle\tilde{v}| = \sum_{i=1}^{M} \frac{1}{\omega^2(C_i)} \left(\sum_{v \in C_i} \omega^2(v) \right) |i\rangle\langle i|, \tag{7.11}$$

But the parenthetic term is simply the square of the weight of cell C_i by definition, leaving

$$Q^T Q = \sum_{i=1}^{M} |i\rangle\langle i| = I_M, \tag{7.12}$$

which completes the proof. □

Lemma 8.1 of Reference [99] provides three statements related to A, Q, and Π, that are mutually equivalent, as well as equivalent to the statement that Π is equitable. The same set of four statements can be shown to be equivalent when A describes a weighted graph, and the equitableness of a partition on a weighted graph is defined as follows.

Definition 7.1. *A partition Π of a weighted undirected connected graph G is* equitable *if, given two cells $C_i, C_j \in \Pi$ and a vertex $u \in C_i$, the quantity*

$$b_{ij}(u) \doteq \sum_{v \in C_j} A_{uv} \frac{\omega(v)}{\omega(u)} \tag{7.13}$$

is a constant b_{ij}, independent of the choice of u.

The above definition is justified by the following theorem, in which a set of vectors is said to be A-invariant if a matrix A maps each of the vectors to a linear combination of one or more of them.

Theorem 7.1. *Suppose $\Pi = \{C_i\}_{i=1}^{M}$ is a partition into M disjoint cells of the vertices of a connected undirected weighted graph G, and Q is its normalized partition matrix. Let A be the adjacency matrix of G. Then the following are equivalent:*

1. Π *is equitable.*

2. *The column space of* Q *is* A*-invariant.*

3. A *and* QQ^T *commute.*

4. *There is an* $M \times M$ *matrix* B *such that* $AQ = QB$.

Proof. The proof consists in showing that each statement implies its successor, cyclically.

(*1* $\implies$ *2*) To show that *1* implies *2*, suppose that Π is equitable and consider column i of Q, denoted

$$|\phi_i\rangle \doteq \sum_{v \in C_i} \Omega(v)|v\rangle. \tag{7.14}$$

The action of A on this column vector is

$$\begin{aligned} A|\phi_i\rangle &= \sum_{v \in C_i} \Omega(v) A|v\rangle = \sum_{v \in C_i} \Omega(v) \sum_{u=1}^{N} A_{uv}|u\rangle \\ &= \frac{1}{\omega(C_i)} \sum_{u=1}^{N} \sum_{v \in C_i} \omega(v) A_{uv}|u\rangle, \end{aligned} \tag{7.15}$$

where the final equality follows from the definition of $\Omega(v)$ Equation 7.8, and the fact that the denominator is constant over the cell C_i. Since Π is equitable by assumption, there exist constants $b_{\tilde{u}i}$ such that

$$\sum_{v \in C_i} \omega(v) A_{uv} = b_{\tilde{u}i} \omega(u), \tag{7.16}$$

so the string of equalities continues as

$$A|\phi_i\rangle = \frac{1}{\omega(C_i)} \sum_{u=1}^{N} b_{\tilde{u}i} \omega(u)|u\rangle = \sum_{j=1}^{M} \frac{b_{ji}}{\omega(C_i)} \sum_{u \in C_j} \omega(u)|u\rangle. \tag{7.17}$$

Finally, defining

$$a_{ij} \doteq b_{ji} \frac{\omega(C_j)}{\omega(C_i)} \tag{7.18}$$

leads to

$$A|\phi_i\rangle = \sum_{j=1}^{M} a_{ij} \sum_{u \in C_j} \Omega(u)|u\rangle = \sum_{j=1}^{M} a_{ij}|\phi_j\rangle. \tag{7.19}$$

That is, A takes each column of Q to a linear combination of the columns of Q; the column space of Q is A-invariant.

(*2* $\implies$ *3*) Now assume that the column space of Q is A-invariant. With the definition (7.14) of the ith column of Q, the partition matrix can be rewritten as

$$Q = \sum_{i=1}^{M} |\phi_i\rangle\langle i|, \tag{7.20}$$

where $|i\rangle$ is M-dimensional and therefore

$$QQ^T = \sum_{i=1}^{M} |\phi_i\rangle\langle i| \sum_{j=1}^{M} |j\rangle\langle\phi_j| = \sum_{i=1}^{M} |\phi_i\rangle\langle\phi_i|. \tag{7.21}$$

The column space of Q is A-invariant by assumption so for each $i \in \{1, \ldots, M\}$,

$$A|\phi_i\rangle = \sum_{j=1}^{M} a_{ij}|\phi_j\rangle \quad \implies \quad \langle\phi_i|A = \sum_{j=1}^{M} a_{ij}\langle\phi_j|, \tag{7.22}$$

since A is real and symmetric, because G is assumed to be undirected. This is a non-trivial statement even though the $|\phi_i\rangle$ form an orthonormal set, since there are only M of them yet they are N-dimensional, and therefore do not form a basis for $\mathbb{R}^N$. Now the commutator of A and QQ^T can be written as

$$\begin{aligned} AQQ^T - QQ^TA &= \sum_{i=1}^{M}\sum_{j=1}^{M} \left[a_{ij}|\phi_j\rangle\langle\phi_i| - a_{ij}|\phi_i\rangle\langle\phi_j|\right] \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} \left(a_{ij} - a_{ji}\right)|\phi_i\rangle\langle\phi_j|, \end{aligned} \tag{7.23}$$

but since the elements of A and the $|\phi_i\rangle$ are real, and A is symmetric,

$$a_{ij} = \langle\phi_j|A|\phi_i\rangle = \langle\phi_i|A|\phi_j\rangle = a_{ji}. \tag{7.24}$$

Therefore the commutator vanishes, as required.

(*3* $\implies$ *4*) To prove that *3* implies *4*, assume now that $[A, QQ^T] = 0$ and consider the matrix $\tilde{B} = Q^T AQ$, from which one sees

$$Q\tilde{B} = QQ^T AQ = AQQ^T Q. \tag{7.25}$$

But Lemma 7.1 shows that $Q^T Q$ is the identity, leaving $Q\tilde{B} = AQ$. Therefore, $\tilde{B}$ is in fact the $m \times m$ matrix B required by the statement.

(*4* $\implies$ *1*) Finally, to see that *4* implies *1*, which will complete the proof, suppose there exists an $m \times m$ matrix B such that $QB = AQ$. Then by definition,

$$\sum_{v=1}^{N} \Omega(v)|v\rangle\langle\tilde{v}|B = \sum_{v=1}^{N} \Omega(v)A|v\rangle\langle\tilde{v}|. \tag{7.26}$$

Multiplying each side by $\langle u|$ on the left and $|j\rangle$ on the right leads to

$$\Omega(u)B_{\tilde{u}j} = \sum_{v=1}^{M} \Omega(v)A_{uv}\delta_{\tilde{v}j} = \sum_{v\in C_j} \Omega(v)A_{uv}. \tag{7.27}$$

From the definition of Ω one obtains

$$\frac{\omega(u)}{\omega(C_{\tilde{u}})}B_{\tilde{u}j} = \frac{1}{\omega(C_j)} \sum_{v\in C_j} \omega(v)A_{uv}, \tag{7.28}$$

which with $i = \tilde{u}$ in turn implies that

$$\sum_{v \in C_j} A_{uv} \frac{\omega(v)}{\omega(u)} = B_{ij} \frac{\omega(C_j)}{\omega(C_i)} \doteq b_{ij}, \tag{7.29}$$

from which the conclusion immediately follows since the final right-hand side is independent of the choice of u from C_i, and therefore Q encodes an equitable partition, Π. □

Statement *3* of Theorem 7.1, that $[A, QQ^T] = 0$, leads to the following useful result.

Corollary 7.1. *Suppose* Π *is an equitable partition of a connected, weighted, undirected graph* G*. Let* A *be the adjacency matrix of* G*, and* Q *the normalized partition matrix of* Π*. Then* QQ^T *has two distinct eigenvalues,* 0 *and* 1*.*

Proof. Since Π is equitable, Theorem 7.1 shows that A and QQ^T commute. Both are real and symmetric, and therefore diagonalizable, thus they are simultaneously diagonalizable and share an eigenspace. Let the eigenvalues of A be λ_i, with corresponding eigenvectors $|\lambda_{i,j}\rangle$ where, for each i, j runs from 1 to the multiplicity of λ_i. Let $q_{i,j}$ be the eigenvalues of QQ^T, so that

$$QQ^T|\lambda_{i,j}\rangle = q_{i,j}|\lambda_{i,j}\rangle. \tag{7.30}$$

Left-multiplying this expression by Q^T yields $Q^T|\lambda_{i,j}\rangle = q_{i,j}Q^T|\lambda_{i,j}\rangle$, from which it can be concluded that either $q_{i,j} = 1$ or $Q^T|\lambda_{i,j}\rangle = 0$. Clearly if the latter is the case, then it is also true that $QQ^T|\lambda_{i,j}\rangle = 0$ which must still equal $q_{i,j}|\lambda_{i,j}\rangle$. But the $|\lambda_{i,j}\rangle$ are non-zero vectors, so in this case $q_{i,j}$ must vanish. Therefore either $q_{i,j} = 1$ or $q_{i,j} = 0$. □

7.2.1 Graph collapse

Given a weighted, connected, undirected graph G and an equitable partition Π of its vertices into M cells, with adjacency and normalized partition matrices A and Q respectively, Theorem 7.1 guarantees that there exists an M-dimensional matrix B such that $AQ = QB$. Lemma 7.1 can then be used to show that

$$B = Q^T AQ. \tag{7.31}$$

B is also a real, symmetric matrix, since

$$\langle i|B|j\rangle = \langle \phi_i|A|\phi_j\rangle = \langle \phi_i|A^T|\phi_j\rangle = \langle \phi_j|A|\phi_i\rangle = \langle j|B|i\rangle. \tag{7.32}$$

Therefore, B can be interpreted as the adjacency matrix of a graph on M vertices; this graph is called the *quotient graph* or *collapsed graph* of G with respect to Π and is denoted G/Π.

Consider now the time evolution operator for a quantum walker on the collapsed graph,

$$U_B \doteq \mathrm{e}^{\mathrm{i}Q^T AQt} = \sum_{k=0}^{\infty} \frac{(\mathrm{i}t)^k}{k!} \left(Q^T AQ\right)^k. \tag{7.33}$$

The $k = 2$ term includes the matrix product

$$\left(Q^T AQ\right)^2 = Q^T AQ\, Q^T AQ = Q^T\, QQ^T AAQ = Q^T A^2 Q, \tag{7.34}$$

where the second step follows from $[A, QQ^T] = 0$ and the third from $Q^T Q = I$. This procedure of commuting QQ^T to the left past any occurrences of A, and then replacing the left-most $Q^T Q$ with the identity is valid for all $k \geq 2$, and in general for integral

$k \geq 0$ each matrix product in the sum of Equation (7.33) can be replaced by

$$\left(Q^T A Q\right)^k = Q^T A^k Q. \tag{7.35}$$

This leads to

$$U_B = Q^T \left(\sum_{k=0}^{\infty} \frac{(it)^k}{k!} A^k \right) Q = Q^T \mathrm{e}^{\mathrm{i}At} Q = Q^T U_A Q, \tag{7.36}$$

which is particularly useful in the case of singleton cells. Suppose vertices u and v of G are singletons in the partition Π. Then the maps from these vertices to their containing cells are invertible: $Q^T|u\rangle = |\tilde{u}\rangle$ and $Q|\tilde{u}\rangle = |u\rangle$. Therefore, the evolution between these vertices on the collapsed graph is identical to that between the associated vertices on the original graph,

$$\langle \tilde{u}|U_B|\tilde{v}\rangle = \langle \tilde{u}|Q^T U_A Q|\tilde{v}\rangle = \langle u|U_A|v\rangle. \tag{7.37}$$

A similar result appears for example in Reference [98] (and references therein) for the case of unweighted graphs. Equation (7.37) as shown above holds for any undirected graph with real edge and self-loop weights under a partition that satisfies the condition for being equitable in the weighted case, Definition 7.1.

This result has further application to computing with quantum walks specifically, and perfect state transfer between subsets more generally. While the time evolutions between singleton cells on G and G/Π are identical because $QQ^T|v\rangle = |v\rangle$ if $|C_{\tilde{v}}| = 1$, QQ^T is not equal to the identity and therefore an arbitrary vertex cannot be mapped from the original graph to the collapsed one and back uniquely. This should not be surprising, since Q^T lowers the dimension of the vertex vectors. Nevertheless, it is the case that QQ^T is block diagonal for a suitable permutation of the vertex labelling, with each block having support on exactly one cell of the partition. Specifically, Equation (7.20) implies

that

$$QQ^T = \sum_{i=1}^{M} |\phi_i\rangle\langle\phi_i|, \tag{7.38}$$

and by definition $\langle v|\phi_i\rangle \neq 0$ if and only if $v \in C_i$. What this means is that the evolution of a quantum walker between vertices of G/Π yields information about the evolution of a walk on G. Defining the superposition $|C_i\rangle \doteq Q|i\rangle$ of vertices in cell i and the 'expanded vector' $|\psi_A\rangle \doteq Q|\psi_B\rangle$ of a state $|\psi_B\rangle$ on G/Π, one obtains

$$\langle C_i|U_A|\psi_A\rangle = \langle i|U_B|\psi_B\rangle. \tag{7.39}$$

For example, consider again the collapse of the cube to the weighted line as in Figure 7.2. There is perfect state transfer between the ends of the line in time $\pi/2$,

$$\left|\langle 4^{(4)}|\mathrm{e}^{\mathrm{i}A_{\mathrm{line}}\pi/2}|1^{(4)}\rangle\right| = 1, \tag{7.40}$$

which translates into PST between the antipodes of the cubes. There is also PST between the other two vertices of the line in the same time, as

$$\left|\langle 3^{(4)}|\mathrm{e}^{\mathrm{i}A_{\mathrm{line}}\pi/2}|2^{(4)}\rangle\right| = 1, \tag{7.41}$$

however these vertices were each collapsed from a cell containing three vertices of the cube so Equation (7.41) does not imply the existence of perfect state transfer between any two vertices on the cube. Nevertheless, it does show that an appropriate superposition over one of the cells transfers perfectly to a superposition of the other cell, according to Equation (7.39). With $|\psi_B\rangle = \alpha|1^{(4)}\rangle + \beta|2^{(4)}\rangle$, let

$$|\psi_A\rangle = Q|\psi_B\rangle = \alpha|1^{(8)}\rangle + \frac{\beta}{\sqrt{3}}\left(|2^{(8)}\rangle + |3^{(8)}\rangle + |4^{(8)}\rangle\right) = \alpha|C_1\rangle + \beta|C_2\rangle. \tag{7.42}$$

Equations (7.40) and (7.41) imply that the state $|\psi_B\rangle$ on the line evolves in time $\pi/2$ to the state $\beta|3^{(4)}\rangle+\alpha|4^{(4)}\rangle$. The state $|\psi_B\rangle$ similarly evolves from a particular superposition of two cells into a superposition of the other two cells,

$$\mathrm{e}^{\mathrm{i}A_{\mathrm{cube}}\pi/2}|\psi_A\rangle = \frac{\beta}{\sqrt{3}}\left(|5^{(8)}\rangle + |6^{(8)}\rangle + |7^{(8)}\rangle\right) + \alpha|8^{(8)}\rangle = \beta|C_3\rangle + \alpha|C_4\rangle. \tag{7.43}$$

This result is particularly useful in the search for multi-walker graphs that effect computational unitaries when only a subset of their vertices encode computational basis states, whether using a PST-based scheme such as the discontinuous-walk model of Chapter 5, or a scheme such as the Bose–Hubbard-based one of Chapter 6 that relies on subset periodicity. Two approaches can be used, applicable to different situations. If, as in Chapter 6, or the case of multiple walkers on P_2 which leads to the hypercube as a secondary graph, a physical system dictates the graphs, then a reduction in the dimensionality of the system through graph collapse can aid in both analytical and numerical analyses. On the other hand, if a particular graph is found to have desirable properties but would be difficult to implement physically, a collapsed version may retain these properties while yielding a smaller, perhaps more easily created graph.

For example, suppose one were to identify that the two bottom vertices of Figure 7.3(a) were of interest, but the remainder of the graph was only relevant inasmuch as it contributed to the behaviour of a walker on those two vertices. (This is the same equitable partitioning seen previously in Figure 7.1(a)). The graph is non-planar, and so does not lend itself well to simulation in a two-dimensional lattice, for example. The sequence of collapses in Figure 7.3 results first in a planar graph with half the number of vertices as the original, and on which the two vertices are still present independently. The second collapse yields a three-vertex graph on which a single vertex corresponds to the pair of interest. This allows the presence or absence of periodicity and perfect state

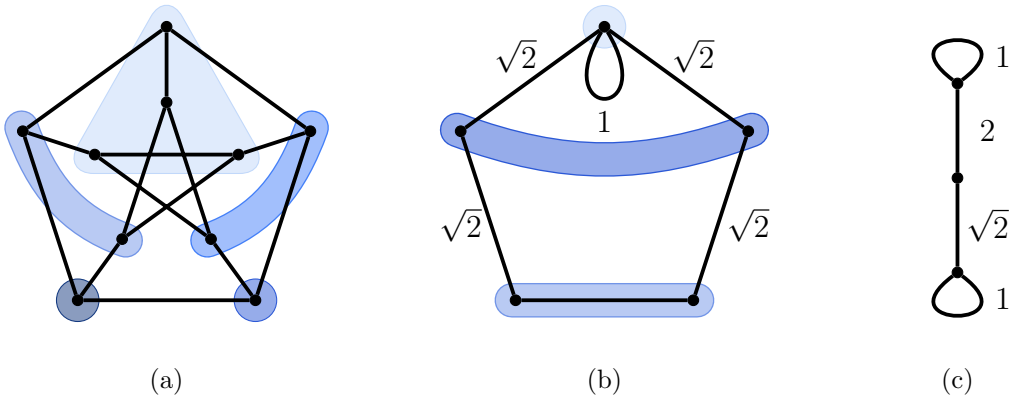


Figure 7.3: Two sequential graph collapses. The equitable partition on the simple graph (a) collapses the graph to (b), introducing weights and a self-loop. The resulting weighted graph itself admits an equitable partition, allowing a further collapse to the graph (c).

transfer on a ten-vertex graph to be analysed in either a five- or three-dimensional space, and shows that the behaviour on the vertices of interest in a non-planar graph can be recreated on a planar graph with the addition of weights and self-loops.

7.2.2 Eigenvalues of collapsed graphs

A remarkable property of the collapsed graph is that it shares all of its eigenvalues with the uncollapsed one. The following lemma makes this statement concrete, and is subsequently used to prove that for a certain large subset of graphs—in particular, all graphs corresponding to bosonic walkers—the Hamiltonians furnished by the collapsed and uncollapsed graphs have equal ground-state energies.

Lemma 7.2. *If $B = Q^T A Q$ is the adjacency matrix of a graph G/Π collapsed under an equitable partition Π from a weighted, undirected, connected graph G, then every eigenvalue of B is also an eigenvalue of A.*

Proof. Let β be an eigenvalue of B, with corresponding eigenvector $|\beta\rangle$. Since Π is

equitable, $AQ = QB$ by Theorem 7.1. Therefore

$$AQ|\beta\rangle = QB|\beta\rangle = \beta Q|\beta\rangle, \tag{7.44}$$

and β is seen to be an eigenvalue of A, as required. □

The proof of Lemma 7.2 additionally shows that the eigenvectors $|\beta\rangle$ of B are related to a subset of the eigenvectors $|\alpha\rangle$ of A by $|\beta\rangle = Q^T|\alpha\rangle$ and $|\alpha\rangle = Q|\beta\rangle$. The determination of which eigenvectors belong to this subset in general remains an open question, though Theorem 7.3 below shows that the eigenvector corresponding to the maximal eigenvalue is preserved under collapse whenever every entry of A is non-negative in the vertex basis. Furthermore, not only does every eigenvector of B yield an eigenvector of A under the action of Q, but every eigenvector of A that does not vanish under the action of Q^T yields an eigenvector of B, as shown by the following lemma.

Lemma 7.3. *Let G be a weighted, undirected, connected graph with adjacency matrix A, and Π an equitable partition with normalized partition matrix Q that generates the quotient graph G/Π with adjacency matrix $B = Q^T AQ$. Suppose $|\lambda_{i,j}\rangle$ is an eigenvector of A as defined in the proof of Corollary 7.1. Then either $Q^T|\lambda_{i,j}\rangle = 0$ or $Q^T|\lambda_{i,j}\rangle$ is an eigenvector of B.*

Proof. Corollary 7.1 shows that the eigenvalues of QQ^T associated with the eigenvectors $|\lambda_{i,j}\rangle$ are $q_{i,j} \in \{0, 1\}$. Suppose first that $QQ^T|\lambda_{i,j}\rangle = 0$. Then since $Q^TQ = I$, left-multiplying this by Q^T yields $Q^T|\lambda_{i,j}\rangle = 0$. On the other hand, in the case $QQ^T|\lambda_{i,j}\rangle = |\lambda_{i,j}\rangle$, one obtains

$$A\left(QQ^T|\lambda_{i,j}\rangle\right) = A|\lambda_{i,j}\rangle = \lambda_i|\lambda_{i,j}\rangle. \tag{7.45}$$

Left-multiplying the initial and final expressions by Q^T yields

$$(Q^T A Q)Q^T|\lambda_{i,j}\rangle = B\left(Q^T|\lambda_{i,j}\rangle\right) = \lambda_i\left(Q^T|\lambda_{i,j}\rangle\right), \tag{7.46}$$

showing that $Q^T|\lambda_{i,j}\rangle$ is an eigenvector of B, with eigenvalue $|\lambda_i\rangle$. □

Some further definitions and results from the fields of linear algebra and graph theory that are useful during the proof of the upcoming Theorem 7.3 are stated here without proof. A treatment can be found, for example, in Reference [101].

Definition 7.2. *An $N \times N$ matrix T is* irreducible *if for each $i, j \in \{1, \ldots, N\}$ there exists a positive integer k such that $(T^k)_{ij} > 0$. A graph on N vertices is irreducible if its adjacency matrix $A(G)$ is irreducible.*

The following lemma relates the irreducibility of a matrix to connectedness of a corresponding unweighted graph.

Lemma 7.4. *An $N \times N$ matrix T with elements T_{ij} is irreducible if and only if the unweighted directed graph Γ_T, defined on the vertex set $\{1, \ldots, N\}$ with an edge from i to j whenever $T_{ij} > 0$, is strongly connected.* □

A graph is *strongly connected* if there is a (directed) path from each vertex to every other vertex. Therefore an undirected graph is strongly connected if and only if it is connected.

Another useful result is the Perron-Frobenius theorem.

Theorem 7.2 (Perron-Frobenius)**.** *Let T be a non-negative irreducible square matrix. Then there exists a real number $\lambda_1 > 0$ with the following properties:*

1. *There exists a real vector $|\lambda_1\rangle$ with each entry strictly positive, and such that $T|\lambda_1\rangle = \lambda_1|\lambda_1\rangle$.*

2. *The algebraic and geometric multiplicities of λ_1 are both equal to 1. That is, its associated eigenspace is one-dimensional.*

3. *For each eigenvalue λ_i of T, $|\lambda_i| \leq \lambda_1$.* □

This completes the prerequisites for the proof of the following theorem.

Theorem 7.3. *Let G be a connected undirected graph with non-negative weights and adjacency matrix A, and let Π be an equitable partition of its vertices with corresponding normalized partition matrix Q. Then the largest eigenvalue of A is unique, and equal to the unique largest eigenvalue of the collapsed graph with adjacency matrix $B = Q^T AQ$.*

Proof. Let the eigenvalues of A be λ_i, with corresponding eigenvectors $|\lambda_{i,j}\rangle$, where for each i, j runs from 1 to the algebraic multiplicity of λ_i. According to Lemma 7.2, each eigenvector of B is given by $Q^T|\lambda_{i,j}\rangle$ for some i and j. The eigenvalues of B are therefore those λ_i for which there exists at least one j such that $Q^T|\lambda_{i,j}\rangle \neq 0$. Consider, then,

$$Q^T|\lambda_{i,j}\rangle = \sum_{v=1}^{N} \Omega(v)|\tilde{v}\rangle\langle v\,|\lambda_{i,j}\rangle = \sum_{k=1}^{M} \left(\sum_{v \in C_k} \Omega(v)\,\langle v\,|\lambda_{i,j}\rangle \right) |k\rangle, \tag{7.47}$$

which shows that $Q^T|\lambda_{i,j}\rangle$ is non-zero if there exists at least one k for which the parenthetic coefficient does not vanish.

Since G is assumed to be connected and undirected, it is strongly connected; it contains no negative edge weights, satisfying the requirements of Lemma 7.4 and the Perron-Frobenius theorem. Therefore there is a unique largest eigenvalue λ_1 of A, with a single corresponding eigenvector $|\lambda_{1,1}\rangle$. The elements $\langle v\,|\lambda_{i,j}\rangle$ of this vector are strictly positive, and by definition $\Omega(v) > 0$ for any vertex with at least one neighbour. Since G is connected, it contains no isolated vertices. As such, for every k the coefficient in Equation (7.47) is a sum of one or more strictly positive

numbers and therefore cannot vanish. Thus $Q^T|\lambda_{1,1}\rangle \neq 0$. So by Lemma 7.3, $Q^T|\lambda_{1,1}\rangle$ is an eigenvector of B, with eigenvalue λ_1. Furthermore, it can be seen from the proof of Lemma 7.3 that λ_1 is the unique largest eigenvalue of B as well as of A. □

7.2.3 Summary of main results

The results of the current section all apply to an arbitrary connected, weighted, undirected graph $G = (V, E, w)$, on $|V| = N$ vertices with adjacency matrix A, and to a partition Π of V into M cells, equitable with respect to G and described by the normalized partition matrix Q. With respect to these quantities, some of the most useful results described above include the following.

First and foremost, Definition 7.1 provides a generalized criterion for the definition of an equitable partition on a weighted graph. Lemma 7.1 shows that Q^TQ is the M-dimensional identity matrix. Theorem 7.1 states not only that A commutes with QQ^T when Π is equitable, but given an adjacency matrix and normalized partition matrix one can determine if the partition is equitable by determining whether these matrices commute. That is, $[A, QQ^T] = 0$ if and only if Π is equitable. An $M \times M$ matrix $B = Q^TAQ$ can be defined that can itself be interpreted as the adjacency matrix of a weighted, undirected graph on M vertices; this graph is G/Π, the quotient graph of G with respect to Π. The probability for a walker to be found in a cell on G evolves identically to the probability to find a walker on the single vertex of G/Π collapsed from that cell. Finally, Lemma 7.2 shows that every eigenvalue of B is an eigenvalue of A, and in particular Theorem 7.3 guarantees that when A contains no negative entries, its largest eigenvalue is preserved under collapse, i.e. is equal to the largest eigenvalue of B.

In particular Theorem 7.3 applies to any system of bosonic walkers on a primary graph with no negative edge weights, i.e. with $\tau_{uv} \geq 0$ in the Bose–Hubbard Hamiltonian for

all vertices $u \neq v$. If the on-site interaction strength g is negative then a constant energy offset can be added to the system, rendering the adjacency matrix irreducible without altering the walkers' dynamics. Note also that Theorem 7.3 does not provide an if-and-only-if result: for adjacency matrices with negative entries, such as for fermionic walkers, the theorem does not apply and the largest eigenvalue may or may not be preserved. That said, no counterexample has yet been identified in which the largest eigenvalue is lost.

7.3 Applications of graph collapse

The full potential of equitable partitioning as it applies to quantum walks is still an active area of ongoing research. To conclude this chapter, a few promising results are presented.

7.3.1 Interacting bosons in periodic potentials

As discussed in Chapter 6, multi-walker quantum walks on graphs are described well by the Bose–Hubbard Hamiltonian, which also plays a large role in condensed-matter physics as a description of many-particle systems confined to lattices [102, 103]. In one dimension such systems can be solved exactly with the Bethe ansatz [104], though in higher dimensions the ground states are in general known only approximately. The fermionic case is of particular interest due to its suspected application to the theory of high-T_c superconductivity [105]. While it can be solved analytically by other means, a system of bosons confined to a one-dimensional ring provides an excellent introduction to the potential applications of weighted graph collapse.

Figure 7.4 shows the secondary graphs corresponding to a system of three interacting indistinguishable bosons hopping on one-dimensional rings of three, four, and five sites. In these cases the number of vertices in the resulting quotient graph is equal to the number of sites on the ring, though this is not a general feature; the collapsed graphs for

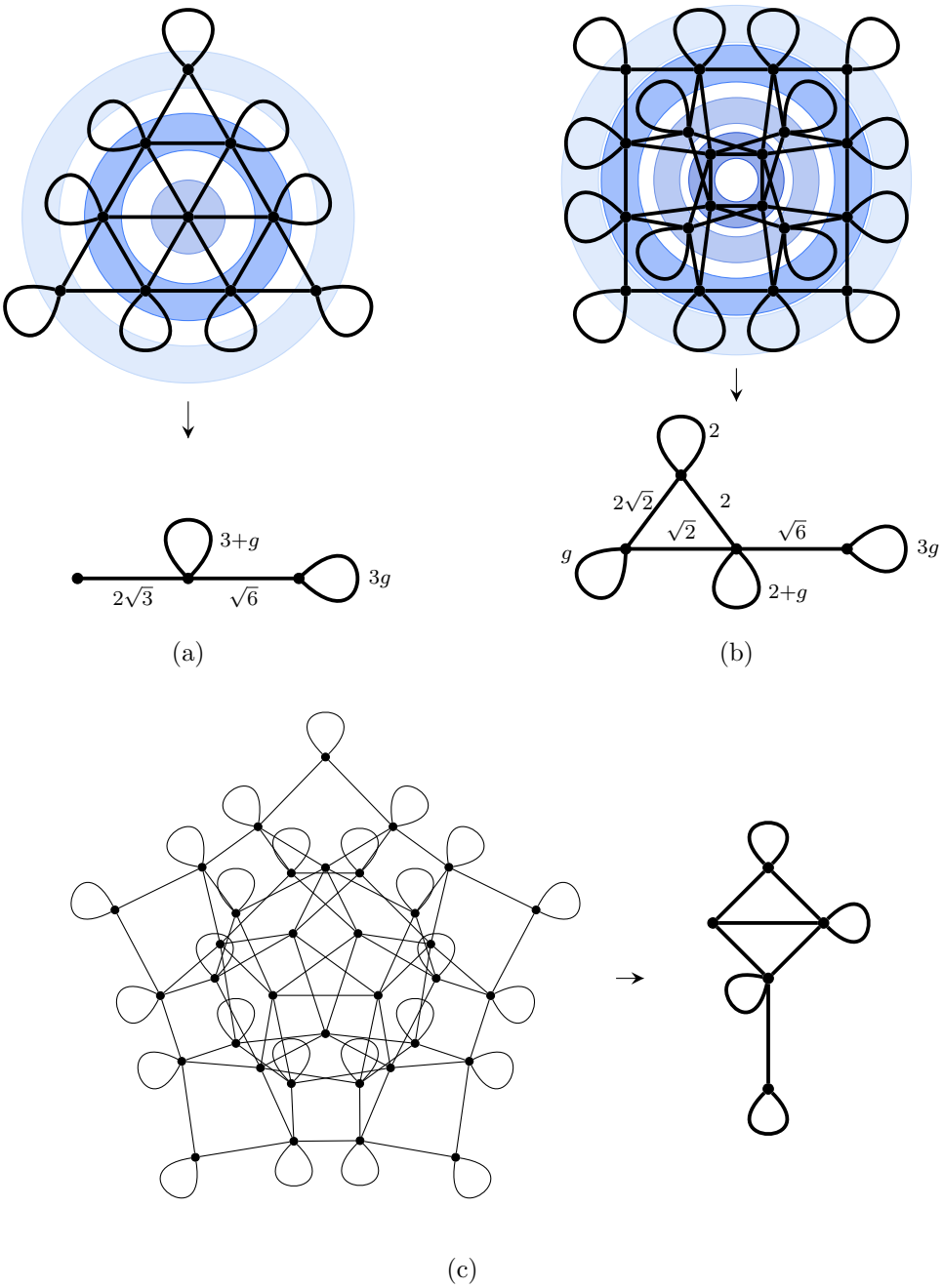


Figure 7.4: Graph collapses for systems of three interacting indistinguishable bosons hopping on one-dimensional rings of (a) three, (b) four, and (c) five sites; g is the on-site interaction strength. All three initial graphs contain multiple values of edge and self-loop weights that have been suppressed for clarity, as does the quotient graph in (c). The partitioning of the original graph for five sites has also been suppressed, though it follows a similar pattern to the other two.

three bosons on rings of six and seven sites have seven and eight vertices, respectively, reduced from 56 and 84 vertices in the uncollapsed secondary graphs. In each case the ground-state energy of the original physical system is given by the negative of the largest eigenvalue of the collapsed graph.

Several variations to this system exist. The bosons could be non-interacting, which is the simplest case to consider as it corresponds to setting $g = 0$. In the other extreme the particles could be hardcore bosons, in the limit $g \to \infty$. When the on-site interaction is so energetically costly, states with multiply occupied sites decouple from the rest, effectively deleting every vertex with a self-loop from the secondary graphs. This does not, however, mean that every vertex with a self-loop can be deleted from the collapsed graphs—Figure 7.4(b) would disappear! Self-loops in collapsed graphs arise whenever the vertices within a given cell are connected to each other, as well as when they are 'connected to themselves' by a self-loop. Finally whether g is zero, infinite, or finite, the particles could also be distinguishable. The same procedure of identifying graph symmetries that yield equitable partitions and then collapsing will allow the determination of the ground-state energy of the system.

7.3.2 Simple graphs via expansion

In general the collapse of simple graphs leads to weighted graphs. One might then wonder whether a given weighted graph can conversely be *expanded* to a simple graph that exhibits the same periodicity and perfect state transfer properties.[1] As of yet there is no general formalism for such an expansion of a graph G by the construction of a larger graph G' such that $G = G'/\Pi$ for some equitable partition Π, but certain heuristics exist that allow such a procedure to be accomplished manually in many cases. For example, consider the collapse shown in Figure 7.5. The graph on $k + 2$ vertices collapses to a

[1]Note that this informal choice of the term 'expansion' as the reverse of the collapse process is unrelated to the formalsim of expander graphs.

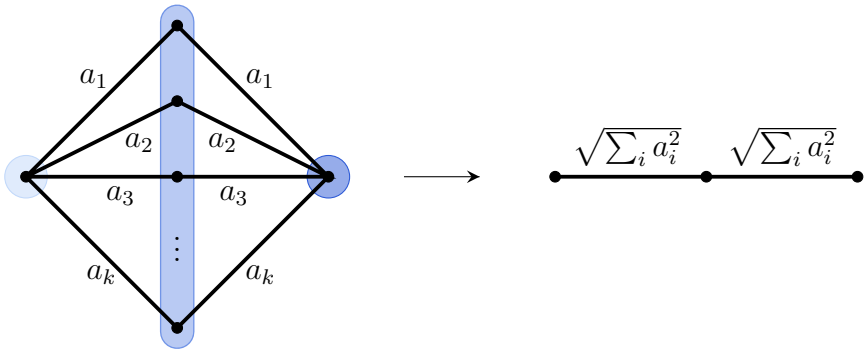


Figure 7.5: Example heuristic for graph expansion. For non-negative weights a_i on each of the k two-segment paths from the left-hand vertex to the right-hand one, the designated partition is equitable and results in a collapse to the two-segment weighted line. Therefore an equally weighted two-segment line can expand to an appropriately weighted set of similar paths between its ends.

line on three vertices. Conversely, given a two-segment line with equal non-negative edge weights b, one can expand it to a graph of the form in Figure 7.5 by choosing k weights a_i such that

$$\sum_{i=1}^{k} a_i^2 \stackrel{!}{=} b^2. \tag{7.48}$$

Note in particular that the expanded graph is planar, and the value of the a_i can be chosen such that the graph has constant weight. Since periodicity and perfect state transfer depend on the adjacency matrix of a graph, multiplied by a scalar time value, the behaviour of a walk on a graph with constant weight is equivalent to that on an unweighted graph in a rescaled time. One application of this is that a graph can be constructed based on a hypercube, with the same distance between two antipodes and such that they exhibit the same periodicity, but with the new graph both planar and simple. The result of this procedure is shown for the four-dimensional hypercube in Figure 7.6.

It is also interesting to note that the quotient graph of a graph with both positive and negative weights can be disconnected, even when the original graph is not. This means

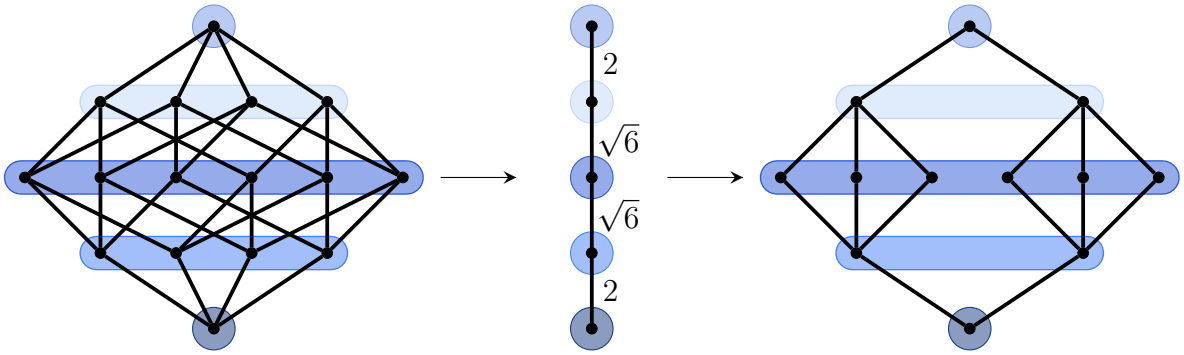


Figure 7.6: Conversion of the four-dimensional hypercube to a planar graph with equivalent evolution on two antipodes, by way of collapse to and expansion from an intermediate weighted line. The simple planar graph on the right exhibits PST from the top vertex to the bottom in time $t = \pi/\sqrt{2}$, while the same evolution on the hypercube occurs in time $t = \pi/2$. To make the evolutions identical, one can give a weight of $\sqrt{2}$ to every edge in the simple graph.

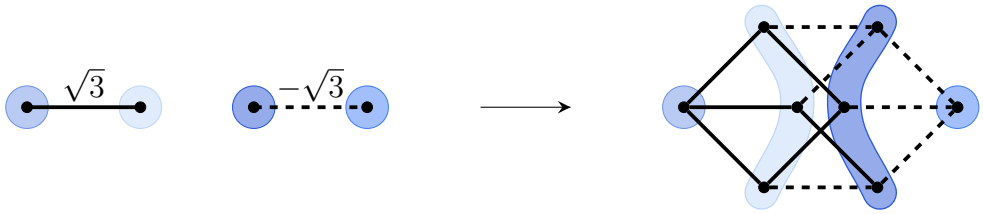


Figure 7.7: Connecting two graphs with a negative-weight expansion. The two copies of K_2 with weights of equal magnitude but opposite sign can be expanded to a connected signed cube. Here dashed edges have negative weights.

that conversely, under certain circumstances edges can be 'expanded' where there were none, by specifying some weights to be negative. For example, two copies of K_2 that are not connected to each other can be expanded to a single connected cube, if the initial graphs have oppositely signed edge weights as shown in Figure 7.7.

Chapter 8

Conclusion

Quantum walks provide a powerful tool for the design of quantum algorithms in the field of quantum computation. They have particular application to problems that can be expressed in terms of graphs, such as the evaluation of decision trees, and the graph isomorphism problem. More recently quantum walks have been shown to be universal for quantum computation, which has been a central theme in this thesis. Under the single-walker continuous-time approach to universal computation, it has been shown that the number of computational unitary operations that can be implemented in a single scattering event grows exponentially with the size of the scattering graph. The catalogue of resulting single-qubit operations has already found application in a new proposal for computing with multiple walkers [69]. The exponential growth with graph size in the number of single-qubit gates that can be performed hints at the possibility of finding multi-qubit gates from a single scattering event.

The discontinuous quantum walk has been proposed as a bridge between the discrete- and continuous-time models. By combining aspects of perfect state transfer with a piecewise-constant Hamiltonian, this hybrid scheme allows a continuous-time quantum walk to deterministically complete a computation as a discrete-time walker can, but without requiring local control over site-dependent coins of multiple dimensions, or indeed any coin degree of freedom at all. The exponential size of the graph makes this scheme unscalable for direct implementation, as is the case with both prior discrete-time and continuous-time schemes. That said, the graph Hamiltonian is easily recast in terms of quantum spin networks, providing a link to possible experimental proofs of concept.

To eliminate the inefficient vertex resource requirements of these schemes, a model

based on multiple walkers interacting under the Bose–Hubbard model has been presented. It makes use of one discontinuous quantum walker for each qubit to be simulated, on a primary graph with only $2n$ vertices. The scheme allows for the execution of arbitrary single-qubit gates through appropriate tunings of local parameters, and can simulate an entangling CPHASE gate at a countably infinite set of phase values. This proves the intuitive conjecture that an increase in the number of walkers should yield a decrease in the number of vertices required for universal quantum computation. In fact the resulting decrease is exponential, making this the first scheme for universal quantum computation based on quantum walks that is in principle scalable. The Bose–Hubbard-based scheme also provides the first quantum walk scheme that allows for the implementation of quantum error correction. The model also has distinctly closer ties to extant experimental techniques, with a straightforward interpretation as bosonic neutral atoms in an optical lattice.

Equitable partitioning is a useful tool for the analysis of multi-walker graphs, as well as of periodicity and perfect state transfer whether there are many walkers present or only one. Previously limited to graphs with edge weights of ± 1, the formalism has been extended herein to arbitrary undirected graphs with real edge weights and self-loops. This allows for sequences of repeated collapse, as quotient graphs of simple graphs are in general weighted. It has also been shown that for a large class of graphs the largest eigenvalue, in this case equal to the spectral radius, peserved under collapse. Thus the ground-state energy of many systems can be determined from much smaller graphs than those present in the given system. In particular all bosonic walks on graphs with non-negative edge weights fall into this class. It is conjectured that this result will hold for undirected fermionic graphs as well, but this remains an open question.

Appendix A

Representative widget information

The set of widget graphs generating the distinct single-qubit unitaries identified in Chapter 4 is contained here in its entirety. The following fields appear in Table A.1:

ID: Unique identifier, assigned in the order the entries appear

N: Number of vertices in the graph

k: Momentum at which this entry results in a unitary

ℓ: Effective length of the graph

Attach.: Zero-based vertex numbers to which tails $\{\underline{0}_{\text{in}}, \underline{1}_{\text{in}}, \underline{0}_{\text{out}}, \underline{1}_{\text{out}}\}$ attach, respectively

(θ, ϕ): Polar and azimuthal angle of the Bloch-sphere axis about which the unitary rotates

α: Angle of the above rotation

Adjacency matrix: $N \times N$ adjacency matrix of the widget graph, written as a list

Equivalent: List of the number of ways to implement the same unitary on $\{N, \ldots, 9\}$ vertices

Since the graphs were searched in order of the number of vertices, no unitary in this list can be implemented at the same momentum and effective length on fewer vertices.

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1	2	4/5	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
2	2	4/5	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
3	2	3/4	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
4	2	3/4	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
5	2	2/3	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
6	2	2/3	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
7	2	3/5	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
8	2	3/5	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
9	2	1/2	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
10	2	1/2	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
11	2	2/5	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
12	2	2/5	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
13	2	1/3	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
14	2	1/3	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
15	2	1/4	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
16	2	1/4	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
17	2	1/5	0	{0, 1, 0, 1}	(0, 0)	0	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
18	2	1/5	0	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00 00	{4, 12, 28, 60, 136, 348, 1184, 6196}
19	4	4/5	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
20	4	4/5	1	{0, 3, 1, 2}	(1.5708, 0)	-3.14159	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
21	4	3/4	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 36, 72, 256, 864}
22	4	3/4	1	{0, 3, 1, 2}	(1.5708, 0)	-3.14159	0010 0001 1000 0100	{8, 8, 36, 72, 256, 864}
23	4	2/3	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 32, 56, 232, 1240}
24	4	2/3	1	{0, 3, 1, 2}	(1.5708, 0)	3.14159	0010 0001 1000 0100	{8, 8, 32, 56, 232, 1240}
25	4	3/5	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
26	4	3/5	1	{0, 3, 1, 2}	(1.5708, 0)	-3.14159	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
27	4	1/2	1	{0, 1, 0, 1}	(0, 0)	0	0001 0001 0001 1110	{36, 76, 444, 2016, 16844, 182896}
28	4	1/2	1	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0001 0001 0001 1110	{36, 76, 444, 2016, 16844, 182896}
29	4	2/5	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
30	4	2/5	1	{0, 3, 1, 2}	(1.5708, 0)	-3.14159	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
31	4	1/3	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 32, 52, 232, 1420}
32	4	1/3	1	{0, 3, 1, 2}	(1.5708, 0)	3.14159	0010 0001 1000 0100	{8, 8, 32, 52, 232, 1420}
33	4	1/4	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 36, 72, 256, 864}
34	4	1/4	1	{0, 3, 1, 2}	(1.5708, 0)	-3.14159	0010 0001 1000 0100	{8, 8, 36, 72, 256, 864}
35	4	1/5	1	{0, 3, 2, 1}	(0, 0)	0	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
36	4	1/5	1	{0, 3, 1, 2}	(1.5708, 0)	3.14159	0010 0001 1000 0100	{8, 8, 32, 48, 144, 360}
37	5	2/3	2	{1, 2, 1, 2}	(0, 0)	0	00011 00001 00001 10000 11100	{12, 44, 288, 2036, 22632}
38	5	2/3	2	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011 00001 00001 10000 11100	{12, 44, 288, 2036, 22632}
39	5	1/2	2	{4, 0, 2, 1}	(0, 0)	1.5708	00011 00011 00001 11000 11100	{4, 4, 80, 377, 4502}
40	5	1/2	2	{0, 4, 1, 2}	(0, 0)	-1.5708	00011 00011 00001 11000 11100	{4, 4, 80, 377, 4502}
41	5	1/2	2	{0, 4, 2, 1}	(1.5708, 0.785398)	-3.14159	00011 00011 00001 11000 11100	{4, 4, 80, 377, 4502}
42	5	1/2	2	{4, 0, 1, 2}	(1.5708, -0.785398)	3.14159	00011 00011 00001 11000 11100	{4, 4, 80, 377, 4502}
43	5	1/2	1	{1, 0, 1, 3}	(0, 0)	1.5708	00010 00001 00001 10000 01100	{6, 26, 96, 467, 2840}
44	5	1/2	1	{0, 1, 3, 1}	(0, 0)	-1.5708	00010 00001 00001 10000 01100	{6, 26, 96, 467, 2840}
45	5	1/2	1	{0, 1, 1, 3}	(1.5708, 0.785398)	-3.14159	00010 00001 00001 10000 01100	{6, 26, 96, 467, 2840}
46	5	1/2	1	{1, 0, 3, 1}	(1.5708, -0.785398)	3.14159	00010 00001 00001 10000 01100	{6, 26, 96, 467, 2840}
47	5	1/2	0.5	{0, 1, 0, 1}	(0, 0)	0	00001 00001 00001 00001 11110	{24, 108, 360, 1908, 12160}
48	5	1/2	0.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001 00001 00001 00001 11110	{24, 108, 360, 1908, 12160}
49	5	1/3	2	{1, 2, 1, 2}	(0, 0)	0	00011 00001 00001 10000 11100	{24, 88, 364, 2076, 18344}
50	5	1/3	2	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011 00001 00001 10000 11100	{24, 88, 364, 2076, 18344}
51	5	1/3	0.5	{1, 2, 1, 2}	(0, 0)	0	00011 00001 00001 10001 11110	{4, 28, 116, 536, 3164}
52	5	1/3	0.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011 00001 00001 10001 11110	{4, 28, 116, 536, 3164}
53	6	4/5	7.23607	{0, 1, 0, 1}	(0, 0)	0	000011 000011 000001 000001 110001 111110	{4, 32, 264, 2496}
54	6	4/5	7.23607	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011 000011 000001 000001 110001 111110	{4, 32, 264, 2496}
55	6	4/5	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
56	6	4/5	2	{0, 3, 2, 1}	(1.5708, 0)	3.14159	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
57	6	3/4	4	{2, 3, 2, 3}	(0, 0)	0	000011 000010 000001 000001 110000 101100	{28, 76, 356, 1784}
58	6	3/4	4	{4, 0, 3, 0}	(0, 0)	-3.14159	000110 000101 000011 110000 101000 011000	{18, 22, 54, 122}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
59	6	3/4	4	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011 000010 000001 000001 110000 101100	{28, 76, 356, 1784}
60	6	3/4	4	{0, 4, 3, 0}	(1.5708, -1.5708)	3.14159	000110 000101 000011 110000 101000 011000	{18, 22, 54, 122}
61	6	3/4	23.3137	{2, 3, 2, 3}	(0, 0)	0	000011 000010 000001 000001 110001 101110	{4, 32, 180, 1064}
62	6	3/4	23.3137	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011 000010 000001 000001 110001 101110	{4, 32, 180, 1064}
63	6	3/4	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{16, 36, 152, 656}
64	6	3/4	2	{0, 1, 5, 4}	(1.5708, 0)	-1.5708	000110 000101 000011 110000 101000 011000	{12, 12, 24, 48}
65	6	3/4	2	{0, 1, 4, 5}	(1.5708, 0)	1.5708	000110 000101 000011 110000 101000 011000	{12, 12, 24, 48}
66	6	3/4	2	{0, 3, 2, 1}	(1.5708, 0)	3.14159	000010 000010 000001 000001 110000 001100	{16, 36, 152, 656}
67	6	3/4	13.6569	{1, 2, 1, 2}	(0, 0)	0	000101 000011 000011 100001 011001 111110	{4, 24, 176, 1244}
68	6	3/4	13.6569	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101 000011 000011 100001 011001 111110	{4, 24, 176, 1244}
69	6	3/4	11.6569	{2, 3, 2, 3}	(0, 0)	0	000011 000011 000001 000001 110001 111110	{4, 24, 112, 504}
70	6	3/4	11.6569	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011 000011 000001 000001 110001 111110	{4, 24, 112, 504}
71	6	2/3	4	{0, 1, 0, 1}	(0, 0)	0	000011 000011 000001 000001 110000 111100	{4, 76, 1190, 27164}
72	6	2/3	4	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011 000011 000001 000001 110000 111100	{4, 76, 1190, 27164}
73	6	2/3	3	{0, 3, 1, 2}	(0, 0)	2.0944	001011 000111 100010 010001 111001 110110	{4, 20, 164, 1307}
74	6	2/3	3	{0, 4, 5, 1}	(0, 0)	0	001011 000111 100010 010001 111000 110100	{8, 96, 576, 4726}
75	6	2/3	3	{3, 0, 2, 1}	(0, 0)	-2.0944	001011 000111 100010 010001 111001 110110	{4, 20, 164, 1307}
76	6	2/3	3	{3, 0, 1, 2}	(1.5708, 1.0472)	-3.14159	001011 000111 100010 010001 111001 110110	{4, 20, 164, 1307}
77	6	2/3	3	{0, 4, 1, 5}	(1.5708, 0)	-3.14159	001011 000111 100010 010001 111000 110100	{8, 96, 576, 4726}
78	6	2/3	3	{0, 3, 2, 1}	(1.5708, -1.0472)	3.14159	001011 000111 100010 010001 111001 110110	{4, 20, 164, 1307}
79	6	2/3	2	{1, 3, 2, 3}	(0, 0)	2.0944	000101 000011 000011 100000 011001 111010	{12, 62, 434, 3692}
80	6	2/3	2	{3, 1, 3, 2}	(0, 0)	-2.0944	000101 000011 000011 100000 011001 111010	{12, 62, 434, 3692}
81	6	2/3	2	{3, 1, 2, 3}	(1.5708, 1.0472)	-3.14159	000101 000011 000011 100000 011001 111010	{12, 62, 434, 3692}
82	6	2/3	2	{1, 3, 3, 2}	(1.5708, -1.0472)	3.14159	000101 000011 000011 100000 011001 111010	{12, 62, 434, 3692}
83	6	2/3	1.5	{1, 2, 1, 2}	(0, 0)	0	000101 000010 000010 100001 011001 100110	{16, 56, 420, 3400}
84	6	2/3	1.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101 000010 000010 100001 011001 100110	{16, 56, 420, 3400}
85	6	3/5	7.23607	{1, 2, 1, 2}	(0, 0)	0	000101 000011 000011 100001 011000 111100	{4, 32, 212, 1480}
86	6	3/5	7.23607	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101 000011 000011 100001 011000 111100	{4, 32, 212, 1480}
87	6	3/5	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
88	6	3/5	2	{0, 3, 2, 1}	(1.5708, 0)	3.14159	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
89	6	1/2	3	{0, 4, 3, 1}	(0, 0)	0	000101 000011 000001 100001 010001 111110	{80, 300, 5694, 60280}
90	6	1/2	3	{2, 0, 1, 0}	(0, 0)	-3.14159	000110 000011 000011 100001 111001 011110	{18, 56, 1291, 10819}
91	6	1/2	3	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	000101 000011 000001 100001 010001 111110	{80, 300, 5694, 60280}
92	6	1/2	3	{0, 2, 1, 0}	(1.5708, -1.5708)	3.14159	000110 000011 000011 100001 111001 011110	{18, 56, 1291, 10819}
93	6	1/2	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{64, 232, 2936, 29828}
94	6	1/2	2	{1, 3, 2, 3}	(0, 0)	-3.14159	000101 000011 000011 100000 011000 111000	{16, 70, 602, 4918}
95	6	1/2	2	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	000010 000010 000001 000001 110000 001100	{64, 232, 2936, 29828}
96	6	1/2	2	{3, 1, 2, 3}	(1.5708, -1.5708)	3.14159	000101 000011 000011 100000 011000 111000	{16, 70, 602, 4918}
97	6	1/2	1.5	{5, 0, 2, 1}	(0, 0)	1.5708	000111 000111 000001 110000 110000 111000	{4, 4, 44, 231}
98	6	1/2	1.5	{0, 5, 1, 2}	(0, 0)	-1.5708	000111 000111 000001 110000 110000 111000	{4, 4, 44, 231}
99	6	1/2	1.5	{0, 5, 2, 1}	(1.5708, 0.785398)	-3.14159	000111 000111 000001 110000 110000 111000	{4, 4, 44, 231}
100	6	1/2	1.5	{5, 0, 1, 2}	(1.5708, -0.785398)	3.14159	000111 000111 000001 110000 110000 111000	{4, 4, 44, 231}
101	6	1/2	0.333333	{0, 1, 0, 1}	(0, 0)	0	000001 000001 000001 000001 000001 111110	{40, 120, 544, 2432}
102	6	1/2	0.333333	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001 000001 000001 000001 000001 111110	{40, 120, 544, 2432}
103	6	2/5	2.76393	{0, 1, 0, 1}	(0, 0)	0	000011 000011 000001 000001 110001 111110	{4, 32, 264, 2496}
104	6	2/5	2.76393	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011 000011 000001 000001 110001 111110	{4, 32, 264, 2496}
105	6	2/5	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
106	6	2/5	2	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
107	6	1/3	4	{0, 1, 0, 1}	(0, 0)	0	000011 000011 000001 000001 110000 111100	{4, 52, 780, 11988}
108	6	1/3	4	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011 000011 000001 000001 110000 111100	{4, 52, 780, 11988}
109	6	1/3	3	{5, 1, 2, 3}	(0, 0)	2.0944	000110 000011 000001 100001 110000 011100	{4, 8, 16, 186}
110	6	1/3	3	{1, 5, 3, 2}	(0, 0)	-2.0944	000110 000011 000001 100001 110000 011100	{4, 8, 16, 186}
111	6	1/3	3	{1, 5, 2, 3}	(1.5708, 1.0472)	-3.14159	000110 000011 000001 100001 110000 011100	{4, 8, 16, 186}
112	6	1/3	3	{5, 1, 3, 2}	(1.5708, -1.0472)	3.14159	000110 000011 000001 100001 110000 011100	{4, 8, 16, 186}
113	6	1/3	2.5	{0, 1, 0, 1}	(0, 0)	0	000111 000111 000011 110000 111001 111010	{4, 60, 508, 4976}
114	6	1/3	2.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000111 000111 000011 110000 111001 111010	{4, 60, 508, 4976}
115	6	1/4	4	{2, 3, 2, 3}	(0, 0)	0	000011 000010 000001 000001 110000 101100	{28, 76, 356, 1784}
116	6	1/4	4	{4, 0, 3, 0}	(0, 0)	-3.14159	000110 000101 000011 110000 101000 011000	{18, 22, 54, 122}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
117	6	1/4	4	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011 000010 000001 000001 110000 101100	{28, 76, 356, 1784}
118	6	1/4	4	{0, 4, 3, 0}	(1.5708, -1.5708)	3.14159	000110 000101 000011 110000 101000 011000	{18, 22, 54, 122}
119	6	1/4	2.34315	{1, 2, 1, 2}	(0, 0)	0	000101 000011 000011 100001 011001 111110	{4, 24, 176, 1244}
120	6	1/4	2.34315	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101 000011 000011 100001 011001 111110	{4, 24, 176, 1244}
121	6	1/4	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{16, 36, 152, 656}
122	6	1/4	2	{0, 1, 4, 5}	(1.5708, 3.14159)	1.5708	000110 000101 000011 110000 101000 011000	{12, 12, 24, 48}
123	6	1/4	2	{0, 1, 5, 4}	(1.5708, 0)	1.5708	000110 000101 000011 110000 101000 011000	{12, 12, 24, 48}
124	6	1/4	2	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	000010 000010 000001 000001 110000 001100	{16, 36, 152, 656}
125	6	1/4	0.686292	{2, 3, 2, 3}	(0, 0)	0	000011 000010 000001 000001 110001 101110	{4, 32, 180, 1064}
126	6	1/4	0.686292	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011 000010 000001 000001 110001 101110	{4, 32, 180, 1064}
127	6	1/4	0.343146	{2, 3, 2, 3}	(0, 0)	0	000011 000011 000001 000001 110001 111110	{4, 24, 112, 504}
128	6	1/4	0.343146	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011 000011 000001 000001 110001 111110	{4, 24, 112, 504}
129	6	1/5	2.76393	{1, 2, 1, 2}	(0, 0)	0	000101 000011 000011 100001 011000 111100	{4, 32, 212, 1480}
130	6	1/5	2.76393	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101 000011 000011 100001 011000 111100	{4, 32, 212, 1480}
131	6	1/5	2	{0, 3, 1, 2}	(0, 0)	0	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
132	6	1/5	2	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	000010 000010 000001 000001 110000 001100	{8, 8, 16, 48}
133	7	4/5	2.76393	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{16, 112, 1196}
134	7	4/5	2.76393	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{16, 112, 1196}
135	7	4/5	18.9443	{2, 3, 2, 3}	(0, 0)	0	0000111 0000010 0000001 0000001 1000001 1100000 1011100	{4, 40, 424}
136	7	4/5	18.9443	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000010 0000001 0000001 1000001 1100000 1011100	{4, 40, 424}
137	7	4/5	14.4721	{5, 6, 5, 6}	(0, 0)	0	0001100 0001011 0000111 1100011 1010011 0111100 0111100	{8, 68, 688}
138	7	4/5	14.4721	{5, 6, 6, 5}	(1.5708, 0)	3.14159	0001100 0001011 0000111 1100011 1010011 0111100 0111100	{8, 68, 688}
139	7	4/5	1.05573	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{4, 40, 420}
140	7	4/5	1.05573	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{4, 40, 420}
141	7	3/4	1	{3, 0, 3, 4}	(0, 0)	2.35619	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
142	7	3/4	1	{0, 3, 4, 3}	(0, 0)	-2.35619	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
143	7	3/4	1	{0, 3, 3, 4}	(1.5708, 1.1781)	-3.14159	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
144	7	3/4	1	{3, 0, 4, 3}	(1.5708, -1.1781)	3.14159	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
145	7	2/3	8	{5, 6, 5, 6}	(0, 0)	0	0000111 0000111 0000011 0000011 1100000 1111000 1111000	{8, 132, 3550}
146	7	2/3	8	{5, 6, 6, 5}	(1.5708, 0)	3.14159	0000111 0000111 0000011 0000011 1100000 1111000 1111000	{8, 132, 3550}
147	7	2/3	6	{1, 2, 1, 2}	(0, 0)	0	0001011 0000110 0000110 1000001 0110001 1110000 1001100	{16, 200, 5252}
148	7	2/3	6	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0001011 0000110 0000110 1000001 0110001 1110000 1001100	{16, 200, 5252}
149	7	2/3	4	{6, 1, 6, 2}	(0, 0)	2.0944	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{4, 239, 4615}
150	7	2/3	4	{0, 5, 2, 5}	(0, 0)	-3.14159	0001111 0000111 0000011 1000100 1101001 1110001 1110110	{4, 124, 1906}
151	7	2/3	4	{1, 6, 2, 6}	(0, 0)	-2.0944	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{4, 239, 4615}
152	7	2/3	4	{1, 6, 6, 2}	(1.5708, 1.0472)	-3.14159	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{4, 239, 4615}
153	7	2/3	4	{5, 0, 2, 5}	(1.5708, -1.5708)	3.14159	0001111 0000111 0000011 1000100 1101001 1110001 1110110	{4, 124, 1906}
154	7	2/3	4	{6, 1, 2, 6}	(1.5708, -1.0472)	3.14159	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{4, 239, 4615}
155	7	2/3	3.5	{1, 2, 1, 2}	(0, 0)	0	0000101 0000011 0000011 0000001 1000001 0110000 1111100	{12, 220, 3418}
156	7	2/3	3.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0000101 0000011 0000011 0000001 1000001 0110000 1111100	{12, 220, 3418}
157	7	2/3	3	{3, 1, 2, 6}	(0, 0)	1.0472	0001011 0000101 0000011 1000000 0100000 1010001 1110010	{12, 108, 906}
158	7	2/3	3	{1, 3, 6, 2}	(0, 0)	-1.0472	0001011 0000101 0000011 1000000 0100000 1010001 1110010	{12, 108, 906}
159	7	2/3	3	{1, 3, 2, 6}	(1.5708, 0.523599)	-3.14159	0001011 0000101 0000011 1000000 0100000 1010001 1110010	{12, 108, 906}
160	7	2/3	3	{3, 1, 6, 2}	(1.5708, -0.523599)	3.14159	0001011 0000101 0000011 1000000 0100000 1010001 1110010	{12, 108, 906}
161	7	2/3	2.5	{0, 3, 1, 2}	(0, 0)	2.0944	0010111 0001111 1000110 0100001 1110011 1110101 1101110	{4, 28, 180}
162	7	2/3	2.5	{0, 6, 5, 1}	(0, 0)	0	0010111 0001011 1000101 0100010 1010001 1101000 1110100	{8, 104, 956}
163	7	2/3	2.5	{3, 0, 2, 1}	(0, 0)	-2.0944	0010111 0001111 1000110 0100001 1110011 1110101 1101110	{4, 28, 180}
164	7	2/3	2.5	{3, 0, 1, 2}	(1.5708, 1.0472)	-3.14159	0010111 0001111 1000110 0100001 1110011 1110101 1101110	{4, 28, 180}
165	7	2/3	2.5	{0, 6, 1, 5}	(1.5708, 0)	-3.14159	0010111 0001011 1000101 0100010 1010001 1101000 1110100	{8, 104, 956}
166	7	2/3	2.5	{0, 3, 2, 1}	(1.5708, -1.0472)	3.14159	0010111 0001111 1000110 0100001 1110011 1110101 1101110	{4, 28, 180}
167	7	2/3	10	{1, 2, 1, 2}	(0, 0)	0	0001101 0000111 0000111 1000010 1110001 0111001 1110110	{4, 56, 1658}
168	7	2/3	10	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0001101 0000111 0000111 1000010 1110001 0111001 1110110	{4, 56, 1658}
169	7	2/3	1.75	{5, 3, 4, 6}	(1.5708, -2.0944)	-1.0472	0001111 0001011 0000111 1100110 1011001 1111001 1110110	{2, 2, 16}
170	7	2/3	1.75	{3, 5, 6, 4}	(1.5708, 2.0944)	-1.0472	0001111 0001011 0000111 1100110 1011001 1111001 1110110	{2, 2, 16}
171	7	2/3	1.75	{3, 5, 4, 6}	(1.10715, 3.14159)	2.63623	0001111 0001011 0000111 1100110 1011001 1111001 1110110	{2, 2, 16}
172	7	2/3	1.75	{5, 3, 6, 4}	(1.10715, 0)	-2.63623	0001111 0001011 0000111 1100110 1011001 1111001 1110110	{2, 2, 16}
173	7	2/3	1.5	{1, 3, 2, 3}	(0, 0)	2.0944	0001001 0000111 0000111 1000000 0110011 0110101 1110110	{8, 46, 268}
174	7	2/3	1.5	{3, 1, 3, 2}	(0, 0)	-2.0944	0001001 0000111 0000111 1000000 0110011 0110101 1110110	{8, 46, 268}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
175	7	2/3	1.5	{3, 1, 2, 3}	(1.5708, 1.0472)	-3.14159	0001001 0000111 0000111 1000000 0110011 0110101 1110110	{8, 46, 268}
176	7	2/3	1.5	{1, 3, 3, 2}	(1.5708, -1.0472)	3.14159	0001001 0000111 0000111 1000000 0110011 0110101 1110110	{8, 46, 268}
177	7	2/3	1.33333	{1, 2, 1, 2}	(0, 0)	0	0001101 0000010 0000010 1000101 1001001 0110001 1001110	{16, 64, 524}
178	7	2/3	1.33333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0001101 0000010 0000010 1000101 1001001 0110001 1001110	{16, 64, 524}
179	7	3/5	5.52786	{0, 1, 0, 1}	(0, 0)	0	0001111 0001111 0000101 1100010 1110001 1101000 1110100	{4, 28, 168}
180	7	3/5	5.52786	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0001111 0001111 0000101 1100010 1110001 1101000 1110100	{4, 28, 168}
181	7	3/5	5.23607	{0, 1, 0, 1}	(0, 0)	0	0000101 0000101 0000011 0000011 1100011 0011101 1111110	{12, 44, 244}
182	7	3/5	5.23607	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0000101 0000101 0000011 0000011 1100011 0011101 1111110	{12, 44, 244}
183	7	3/5	49.5967	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000000 1100001 1111010	{4, 32, 172}
184	7	3/5	49.5967	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000000 1100001 1111010	{4, 32, 172}
185	7	3/5	4.73607	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000001 1100001 1111110	{4, 24, 108}
186	7	3/5	4.73607	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000001 1100001 1111110	{4, 24, 108}
187	7	3/5	26.1803	{0, 1, 0, 1}	(0, 0)	0	0000101 0000101 0000011 0000011 1100001 0011000 1111100	{4, 24, 148}
188	7	3/5	26.1803	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0000101 0000101 0000011 0000011 1100001 0011000 1111100	{4, 24, 148}
189	7	3/5	2.76393	{2, 3, 2, 3}	(0, 0)	0	0000110 0000011 0000001 0000001 1000000 1100000 0111000	{32, 80, 336}
190	7	3/5	2.76393	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000110 0000011 0000001 0000001 1000000 1100000 0111000	{32, 80, 336}
191	7	3/5	1.80902	{2, 3, 2, 3}	(0, 0)	0	0000111 0000010 0000001 0000001 1000000 1100001 1011010	{4, 24, 108}
192	7	3/5	1.80902	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000010 0000001 0000001 1000000 1100001 1011010	{4, 24, 108}
193	7	3/5	1.44721	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{4, 32, 172}
194	7	3/5	1.44721	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{4, 32, 172}
195	7	3/5	0.690983	{2, 3, 2, 3}	(0, 0)	0	0000110 0000011 0000001 0000001 1000001 1100000 0111100	{4, 24, 108}
196	7	3/5	0.690983	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000110 0000011 0000001 0000001 1000001 1100000 0111100	{4, 24, 108}
197	7	1/2	5	{0, 5, 2, 1}	(0, 0)	1.5708	0001110 0000111 0000011 1000011 1100001 1111000 0111100	{24, 60, 15439}
198	7	1/2	5	{5, 0, 1, 2}	(0, 0)	-1.5708	0001110 0000111 0000011 1000011 1100001 1111000 0111100	{24, 60, 15439}
199	7	1/2	5	{5, 0, 2, 1}	(1.5708, 0.785398)	-3.14159	0001110 0000111 0000011 1000011 1100001 1111000 0111100	{24, 60, 15439}
200	7	1/2	5	{0, 5, 1, 2}	(1.5708, -0.785398)	3.14159	0001110 0000111 0000011 1000011 1100001 1111000 0111100	{24, 60, 15439}
201	7	1/2	4	{5, 0, 2, 1}	(0, 0)	1.5708	0000111 0000110 0000011 0000001 1100001 1110000 1011100	{32, 112, 14451}
202	7	1/2	4	{0, 5, 1, 2}	(0, 0)	-1.5708	0000111 0000110 0000011 0000001 1100001 1110000 1011100	{32, 112, 14451}
203	7	1/2	4	{0, 5, 2, 1}	(1.5708, 0.785398)	-3.14159	0000111 0000110 0000011 0000001 1100001 1110000 1011100	{32, 112, 14451}
204	7	1/2	4	{5, 0, 1, 2}	(1.5708, -0.785398)	3.14159	0000111 0000110 0000011 0000001 1100001 1110000 1011100	{32, 112, 14451}
205	7	1/2	3	{3, 1, 2, 4}	(0, 0)	1.5708	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{188, 967, 22009}
206	7	1/2	3	{1, 3, 4, 2}	(0, 0)	-1.5708	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{188, 967, 22009}
207	7	1/2	3	{1, 3, 2, 4}	(1.5708, 0.785398)	-3.14159	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{188, 967, 22009}
208	7	1/2	3	{3, 1, 4, 2}	(1.5708, -0.785398)	3.14159	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{188, 967, 22009}
209	7	1/2	2.5	{0, 1, 0, 1}	(0, 0)	0	0000111 0000111 0000011 0000011 1100001 1111000 1111100	{56, 344, 7160}
210	7	1/2	2.5	{0, 5, 1, 5}	(0, 0)	-3.14159	0000101 0000101 0000011 0000011 1100011 0011100 1111100	{14, 44, 1445}
211	7	1/2	2.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0000111 0000111 0000011 0000011 1100001 1111000 1111100	{56, 344, 7160}
212	7	1/2	2.5	{5, 0, 1, 5}	(1.5708, -1.5708)	3.14159	0000101 0000101 0000011 0000011 1100011 0011100 1111100	{14, 44, 1445}
213	7	1/2	1.66667	{0, 3, 1, 2}	(0, 0)	0	0000101 0000101 0000011 0000011 1100000 0011000 1111000	{24, 112, 3032}
214	7	1/2	1.66667	{2, 0, 1, 0}	(0, 0)	-3.14159	0001010 0000111 0000111 1000001 0110000 1110001 0111010	{14, 44, 746}
215	7	1/2	1.66667	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	0000101 0000101 0000011 0000011 1100000 0011000 1111000	{24, 112, 3032}
216	7	1/2	1.66667	{0, 2, 1, 0}	(1.5708, -1.5708)	3.14159	0001010 0000111 0000111 1000001 0110000 1110001 0111010	{14, 44, 746}
217	7	1/2	1.5	{1, 2, 1, 2}	(0, 0)	0	0000011 0000010 0000010 0000001 0000001 1110000 1001100	{56, 324, 3252}
218	7	1/2	1.5	{5, 0, 4, 0}	(0, 0)	-3.14159	0000111 0000110 0000110 0000001 1110001 1110001 1001110	{12, 40, 350}
219	7	1/2	1.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0000011 0000010 0000010 0000001 0000001 1110000 1001100	{56, 324, 3252}
220	7	1/2	1.5	{0, 5, 4, 0}	(1.5708, -1.5708)	3.14159	0000111 0000110 0000110 0000001 1110001 1110001 1001110	{12, 40, 350}
221	7	1/2	1.33333	{0, 6, 4, 5}	(0, 0)	1.5708	0000111 0000011 0000011 0000011 1000000 1111000 1111000	{4, 4, 44}
222	7	1/2	1.33333	{6, 0, 5, 4}	(0, 0)	-1.5708	0000111 0000011 0000011 0000011 1000000 1111000 1111000	{4, 4, 44}
223	7	1/2	1.33333	{6, 0, 4, 5}	(1.5708, 0.785398)	-3.14159	0000111 0000011 0000011 0000011 1000000 1111000 1111000	{4, 4, 44}
224	7	1/2	1.33333	{0, 6, 5, 4}	(1.5708, -0.785398)	3.14159	0000111 0000011 0000011 0000011 1000000 1111000 1111000	{4, 4, 44}
225	7	1/2	0.75	{1, 2, 1, 2}	(0, 0)	0	0000111 0000011 0000011 0000011 1000001 1111000 1111100	{88, 588, 8748}
226	7	1/2	0.75	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0000111 0000011 0000011 0000011 1000001 1111000 1111100	{88, 588, 8748}
227	7	1/2	0.666667	{2, 3, 2, 3}	(0, 0)	0	0000011 0000010 0000001 0000001 0000001 1100000 1011100	{112, 592, 6124}
228	7	1/2	0.666667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000011 0000010 0000001 0000001 0000001 1100000 1011100	{112, 592, 6124}
229	7	1/2	0.25	{0, 1, 0, 1}	(0, 0)	0	0000001 0000001 0000001 0000001 0000001 0000001 1111110	{60, 180, 720}
230	7	1/2	0.25	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0000001 0000001 0000001 0000001 0000001 0000001 1111110	{60, 180, 720}
231	7	2/5	7.23607	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{16, 112, 1196}
232	7	2/5	7.23607	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{16, 112, 1196}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
233	7	2/5	5.52786	{5, 6, 5, 6}	(0, 0)	0	0001100 0001011 0000111 1100011 1010011 0111100 0111100	{8, 68, 688}
234	7	2/5	5.52786	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	0001100 0001011 0000111 1100011 1010011 0111100 0111100	{8, 68, 688}
235	7	2/5	18.9443	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{4, 40, 420}
236	7	2/5	18.9443	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{4, 40, 420}
237	7	2/5	1.05573	{2, 3, 2, 3}	(0, 0)	0	0000111 0000010 0000001 0000001 1000001 1100000 1011100	{4, 40, 424}
238	7	2/5	1.05573	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000010 0000001 0000001 1000001 1100000 1011100	{4, 40, 424}
239	7	1/3	8	{5, 6, 5, 6}	(0, 0)	0	0000111 0000111 0000011 0000011 1100000 1111000 1111000	{8, 158, 2980}
240	7	1/3	8	{5, 6, 6, 5}	(1.5708, 0)	3.14159	0000111 0000111 0000011 0000011 1100000 1111000 1111000	{8, 158, 2980}
241	7	1/3	7	{0, 4, 3, 1}	(0, 0)	0	0001001 0000101 0000011 1000001 0100001 0010000 1111100	{16, 88, 496}
242	7	1/3	7	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	0001001 0000101 0000011 1000001 0100001 0010000 1111100	{16, 88, 496}
243	7	1/3	4	{1, 6, 2, 6}	(0, 0)	2.0944	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{6, 126, 1662}
244	7	1/3	4	{3, 0, 2, 6}	(0, 0)	1.0472	0000110 0000100 0000011 0000011 1100001 1011000 0011100	{4, 20, 70}
245	7	1/3	4	{6, 1, 6, 2}	(0, 0)	-2.0944	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{6, 126, 1662}
246	7	1/3	4	{0, 3, 6, 2}	(0, 0)	-1.0472	0000110 0000100 0000011 0000011 1100001 1011000 0011100	{4, 20, 70}
247	7	1/3	4	{6, 1, 2, 6}	(1.5708, 1.0472)	-3.14159	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{6, 126, 1662}
248	7	1/3	4	{0, 3, 2, 6}	(1.5708, 0.523599)	-3.14159	0000110 0000100 0000011 0000011 1100001 1011000 0011100	{4, 20, 70}
249	7	1/3	4	{1, 6, 6, 2}	(1.5708, -1.0472)	3.14159	0000111 0000101 0000011 0000001 1100000 1010000 1111000	{6, 126, 1662}
250	7	1/3	4	{3, 0, 6, 2}	(1.5708, -0.523599)	3.14159	0000110 0000100 0000011 0000011 1100001 1011000 0011100	{4, 20, 70}
251	7	1/3	3	{1, 6, 4, 3}	(0, 0)	0	0000110 0000100 0000011 0000001 1100001 1010000 0011100	{32, 220, 1424}
252	7	1/3	3	{1, 6, 3, 4}	(1.5708, 0)	3.14159	0000110 0000100 0000011 0000001 1100001 1010000 0011100	{32, 220, 1424}
253	7	1/3	2.5	{0, 6, 3, 5}	(0, 0)	2.0944	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 48, 272}
254	7	1/3	2.5	{6, 0, 6, 3}	(0, 0)	1.0472	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 92, 616}
255	7	1/3	2.5	{6, 0, 5, 3}	(0, 0)	-2.0944	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 48, 272}
256	7	1/3	2.5	{0, 6, 3, 6}	(0, 0)	-1.0472	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 92, 616}
257	7	1/3	2.5	{6, 0, 3, 5}	(1.5708, 1.0472)	-3.14159	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 48, 272}
258	7	1/3	2.5	{0, 6, 6, 3}	(1.5708, 0.523599)	-3.14159	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 92, 616}
259	7	1/3	2.5	{0, 6, 5, 3}	(1.5708, -1.0472)	3.14159	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 48, 272}
260	7	1/3	2.5	{6, 0, 3, 6}	(1.5708, -0.523599)	3.14159	0001011 0000111 0000001 1000011 0100011 1101100 1111100	{12, 92, 616}
261	7	1/3	2	{3, 1, 3, 2}	(0, 0)	2.0944	0000101 0000010 0000010 0000001 1000000 0110000 1001000	{8, 44, 161}
262	7	1/3	2	{1, 3, 2, 3}	(0, 0)	-2.0944	0000101 0000010 0000010 0000001 1000000 0110000 1001000	{8, 44, 161}
263	7	1/3	2	{1, 3, 3, 2}	(1.5708, 1.0472)	-3.14159	0000101 0000010 0000010 0000001 1000000 0110000 1001000	{8, 44, 161}
264	7	1/3	2	{3, 1, 2, 3}	(1.5708, -1.0472)	3.14159	0000101 0000010 0000010 0000001 1000000 0110000 1001000	{8, 44, 161}
265	7	1/3	1.75	{1, 2, 1, 2}	(0, 0)	0	0000110 0000011 0000011 0000001 1000001 1110000 0111100	{8, 88, 916}
266	7	1/3	1.75	{4, 0, 3, 5}	(1.5708, -2.0944)	-1.0472	0001101 0000111 0000111 1000011 1110000 0111000 1111000	{2, 4, 12}
267	7	1/3	1.75	{0, 4, 5, 3}	(1.5708, 2.0944)	-1.0472	0001101 0000111 0000111 1000011 1110000 0111000 1111000	{2, 4, 12}
268	7	1/3	1.75	{0, 4, 3, 5}	(1.10715, 3.14159)	2.63623	0001101 0000111 0000111 1000011 1110000 0111000 1111000	{2, 4, 12}
269	7	1/3	1.75	{4, 0, 5, 3}	(1.10715, 0)	-2.63623	0001101 0000111 0000111 1000011 1110000 0111000 1111000	{2, 4, 12}
270	7	1/3	1.75	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0000110 0000011 0000011 0000001 1000001 1110000 0111100	{8, 88, 916}
271	7	1/3	1.6	{1, 2, 1, 2}	(0, 0)	0	0001101 0000111 0000111 1000010 1110001 0111000 1110100	{4, 36, 232}
272	7	1/3	1.6	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	0001101 0000111 0000111 1000010 1110001 0111000 1110100	{4, 36, 232}
273	7	1/3	1.5	{1, 5, 4, 2}	(0, 0)	0	0001001 0000101 0000011 1000001 0100000 0010000 1111000	{8, 52, 520}
274	7	1/3	1.5	{1, 5, 2, 4}	(1.5708, 0)	3.14159	0001001 0000101 0000011 1000001 0100000 0010000 1111000	{8, 52, 520}
275	7	1/3	0.4	{2, 3, 2, 3}	(0, 0)	0	0000101 0000011 0000001 0000001 1000001 0100000 1111100	{4, 32, 156}
276	7	1/3	0.4	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000101 0000011 0000001 0000001 1000001 0100000 1111100	{4, 32, 156}
277	7	1/3	0.25	{2, 3, 2, 3}	(0, 0)	0	0000101 0000011 0000001 0000001 1000001 0100001 1111110	{4, 24, 100}
278	7	1/3	0.25	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000101 0000011 0000001 0000001 1000001 0100001 1111110	{4, 24, 100}
279	7	1/4	1	{3, 0, 3, 4}	(0, 0)	0.785398	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
280	7	1/4	1	{0, 3, 4, 3}	(0, 0)	-0.785398	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
281	7	1/4	1	{0, 3, 3, 4}	(1.5708, 0.392699)	-3.14159	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
282	7	1/4	1	{3, 0, 4, 3}	(1.5708, -0.392699)	3.14159	0000100 0000011 0000011 0000001 1000000 0110000 0111000	{3, 19, 71}
283	7	1/5	7.23607	{2, 3, 2, 3}	(0, 0)	0	0000110 0000011 0000001 0000001 1000000 1100000 0111000	{32, 80, 336}
284	7	1/5	7.23607	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000110 0000011 0000001 0000001 1000000 1100000 0111000	{32, 80, 336}
285	7	1/5	3.81966	{0, 1, 0, 1}	(0, 0)	0	0000101 0000101 0000011 0000011 1100001 0011000 1111100	{4, 24, 148}
286	7	1/5	3.81966	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0000101 0000101 0000011 0000011 1100001 0011000 1111100	{4, 24, 148}
287	7	1/5	14.4721	{0, 1, 0, 1}	(0, 0)	0	0001111 0001111 0000101 1100010 1110001 1101000 1110100	{4, 28, 168}
288	7	1/5	14.4721	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0001111 0001111 0000101 1100010 1110001 1101000 1110100	{4, 28, 168}
289	7	1/5	1.80902	{2, 3, 2, 3}	(0, 0)	0	0000110 0000011 0000001 0000001 1000001 1100000 0111100	{4, 24, 108}
290	7	1/5	1.80902	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000110 0000011 0000001 0000001 1000001 1100000 0111100	{4, 24, 108}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
291	7	1/5	1.05573	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{8, 88, 656}
292	7	1/5	1.05573	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000000 1100000 1111000	{8, 88, 656}
293	7	1/5	0.763932	{0, 1, 0, 1}	(0, 0)	0	0000101 0000101 0000011 0000011 1100011 0011101 1111110	{12, 44, 244}
294	7	1/5	0.763932	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	0000101 0000101 0000011 0000011 1100011 0011101 1111110	{12, 44, 244}
295	7	1/5	0.690983	{2, 3, 2, 3}	(0, 0)	0	0000111 0000010 0000001 0000001 1000000 1100001 1011010	{4, 24, 108}
296	7	1/5	0.690983	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000010 0000001 0000001 1000000 1100001 1011010	{4, 24, 108}
297	7	1/5	0.552786	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{4, 32, 172}
298	7	1/5	0.552786	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000001 1100000 1111100	{4, 32, 172}
299	7	1/5	0.403252	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000000 1100001 1111010	{4, 32, 172}
300	7	1/5	0.403252	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000000 1100001 1111010	{4, 32, 172}
301	7	1/5	0.263932	{2, 3, 2, 3}	(0, 0)	0	0000111 0000011 0000001 0000001 1000001 1100001 1111110	{4, 24, 108}
302	7	1/5	0.263932	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	0000111 0000011 0000001 0000001 1000001 1100001 1111110	{4, 24, 108}
303	8	4/5	99.1935	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000101 10000011 11110001 11101001 11111110	{4, 44}
304	8	4/5	99.1935	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000101 10000011 11110001 11101001 11111110	{4, 44}
305	8	4/5	8.2918	{1, 2, 1, 2}	(0, 0)	0	00001011 00000101 00000101 00000010 10000011 01100000 10011000 11101000	{12, 116}
306	8	4/5	8.2918	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001011 00000101 00000101 00000010 10000011 01100000 10011000 11101000	{12, 116}
307	8	4/5	8.23607	{1, 6, 5, 2}	(0, 0)	0	00000111 00000100 00000010 00000001 00000001 11000001 10100001 10011110	{8, 56}
308	8	4/5	8.23607	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	00000111 00000100 00000010 00000001 00000001 11000001 10100001 10011110	{8, 56}
309	8	4/5	68.541	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00001011 00000111 11100001 11010001 11110001 11111110	{4, 60}
310	8	4/5	68.541	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001111 00001111 00001011 00000111 11100001 11010001 11110001 11111110	{4, 60}
311	8	4/5	6.54509	{2, 3, 2, 3}	(0, 0)	0	00001101 00000110 00000001 00000001 10000011 11000000 01001000 10111000	{8, 64}
312	8	4/5	6.54509	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001101 00000110 00000001 00000001 10000011 11000000 01001000 10111000	{8, 64}
313	8	4/5	56.8328	{5, 6, 5, 6}	(0, 0)	0	00011111 00011111 00000110 11001111 11010111 11111001 11111001 11011110	{4, 48}
314	8	4/5	56.8328	{5, 6, 6, 5}	(1.5708, 0)	3.14159	00011111 00011111 00000110 11001111 11010111 11111001 11111001 11011110	{4, 48}
315	8	4/5	5.52786	{2, 3, 2, 3}	(0, 0)	0	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{24, 432}
316	8	4/5	5.52786	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{24, 432}
317	8	4/5	5.42705	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000010 00000001 11000011 11110101 11101110	{8, 88}
318	8	4/5	5.42705	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000010 00000001 11000011 11110101 11101110	{8, 88}
319	8	4/5	37.8885	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{12, 232}
320	8	4/5	37.8885	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{12, 232}
321	8	4/5	3.81966	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00000100 00000010 11000001 11100001 11010001 11001110	{8, 84}
322	8	4/5	3.81966	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001111 00001111 00000100 00000010 11000001 11100001 11010001 11001110	{8, 84}
323	8	4/5	3.76393	{0, 5, 4, 1}	(0, 0)	0	00001011 00000111 00000001 00000001 10000011 01000011 11001101 11111110	{8, 80}
324	8	4/5	3.76393	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001011 00000111 00000001 00000001 10000011 01000011 11001101 11111110	{8, 80}
325	8	4/5	3.16718	{5, 6, 5, 6}	(0, 0)	0	00001111 00000111 00000111 00000111 10000111 11111001 11111001 11111110	{4, 16}
326	8	4/5	3.16718	{5, 6, 6, 5}	(1.5708, 0)	3.14159	00001111 00000111 00000111 00000111 10000111 11111001 11111001 11111110	{4, 16}
327	8	4/5	3	{2, 5, 4, 3}	(0, 0)	0	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 8}
328	8	4/5	3	{2, 5, 3, 4}	(1.5708, 0)	-3.14159	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 8}
329	8	4/5	26.1803	{5, 6, 5, 6}	(0, 0)	0	00011111 00001111 00000110 10001111 11010111 11111001 11111001 11011110	{4, 108}
330	8	4/5	26.1803	{5, 6, 6, 5}	(1.5708, 0)	3.14159	00011111 00001111 00000110 10001111 11010111 11111001 11111001 11011110	{4, 108}
331	8	4/5	21.7082	{6, 7, 6, 7}	(0, 0)	0	00000111 00000111 00000111 00000011 00000011 11100011 11111100 11111100	{4, 40}
332	8	4/5	21.7082	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	00000111 00000111 00000111 00000011 00000011 11100011 11111100 11111100	{4, 40}
333	8	4/5	2.89443	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 36}
334	8	4/5	2.89443	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 36}
335	8	4/5	2.5	{2, 3, 2, 3}	(0, 0)	0	00001100 00000101 00000010 00000010 10000001 11000000 00110001 01001010	{8, 64}
336	8	4/5	2.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001100 00000101 00000010 00000010 10000001 11000000 00110001 01001010	{8, 64}
337	8	4/5	148.79	{2, 3, 2, 3}	(0, 0)	0	00001011 00000111 00000101 00000101 10000011 01110010 11001101 11111010	{4, 32}
338	8	4/5	148.79	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000111 00000101 00000101 10000011 01110010 11001101 11111010	{4, 32}
339	8	4/5	10	{1, 2, 1, 2}	(0, 0)	0	00000110 00000101 00000101 00000011 00000011 11100001 10011001 01111110	{24, 320}
340	8	4/5	10	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000110 00000101 00000101 00000011 00000011 11100001 10011001 01111110	{24, 320}
341	8	4/5	1.65836	{2, 3, 2, 3}	(0, 0)	0	00001011 00000111 00000101 00000101 10000011 01110000 11001000 11111000	{4, 32}
342	8	4/5	1.65836	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000111 00000101 00000101 10000011 01110000 11001000 11111000	{4, 32}
343	8	4/5	0.954915	{2, 3, 2, 3}	(0, 0)	0	00001101 00000111 00000001 00000001 10000010 11000000 01001000 11110000	{8, 64}
344	8	4/5	0.954915	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001101 00000111 00000001 00000001 10000010 11000000 01001000 11110000	{8, 64}
345	8	3/4	91.2548	{1, 2, 1, 2}	(0, 0)	0	00011111 00001111 00001111 10000111 11100000 11110001 11110001 11110110	{4, 28}
346	8	3/4	91.2548	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011111 00001111 00001111 10000111 11100000 11110001 11110001 11110110	{4, 28}
347	8	3/4	9	{0, 4, 3, 1}	(0, 0)	0	00010011 00001011 00000101 10000011 01000011 00100001 11011001 11111110	{8, 32}
348	8	3/4	9	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	00010011 00001011 00000101 10000011 01000011 00100001 11011001 11111110	{8, 32}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
349	8	3/4	8.68629	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000101 00000011 00000010 11100001 11011000 11110100	{4, 40}
350	8	3/4	8.68629	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000101 00000011 00000010 11100001 11011000 11110100	{4, 40}
351	8	3/4	8.34315	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00001101 00000011 11100000 11100000 11010001 11110010	{4, 28}
352	8	3/4	8.34315	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001111 00001111 00001101 00000011 11100000 11100000 11010001 11110010	{4, 28}
353	8	3/4	8	{0, 1, 0, 1}	(0, 0)	0	00000011 00000011 00000010 00000001 00000001 00000001 11100000 11011100	{24, 332}
354	8	3/4	8	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000011 00000011 00000010 00000001 00000001 00000001 11100000 11011100	{24, 332}
355	8	3/4	7.82843	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000011 10000001 01000001 00110001 11111110	{12, 72}
356	8	3/4	7.82843	{0, 1, 3, 2}	(1.5708, -3.14159)	1.5708	00001101 00001011 00000101 00000011 11000000 10100001 01010001 11110110	{4, 8}
357	8	3/4	7.82843	{0, 1, 2, 3}	(1.5708, 3.14159)	-1.5708	00001101 00001011 00000101 00000011 11000000 10100001 01010001 11110110	{4, 8}
358	8	3/4	7.82843	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000011 10000001 01000001 00110001 11111110	{12, 72}
359	8	3/4	7.02944	{5, 6, 5, 6}	(0, 0)	0	00101111 00010111 10001111 01000111 10100111 11111001 11111001 11111110	{4, 20}
360	8	3/4	7.02944	{5, 6, 6, 5}	(1.5708, 0)	3.14159	00101111 00010111 10001111 01000111 10100111 11111001 11111001 11111110	{4, 20}
361	8	3/4	69.9411	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000010 00000001 10000001 11110000 11101100	{4, 56}
362	8	3/4	69.9411	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000010 00000001 10000001 11110000 11101100	{4, 56}
363	8	3/4	6.82843	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000101 10000011 11110000 11101001 11111010	{12, 108}
364	8	3/4	6.82843	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000101 10000011 11110000 11101001 11111010	{12, 108}
365	8	3/4	6	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000001 00000001 10000010 11100100 11111000	{20, 196}
366	8	3/4	6	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000001 00000001 10000010 11100100 11111000	{20, 196}
367	8	3/4	58.2843	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000001 10000001 11100000 11100000 11111000	{8, 52}
368	8	3/4	58.2843	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000001 10000001 11100000 11100000 11111000	{8, 52}
369	8	3/4	50.6274	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000101 00000010 00000001 11100001 11010000 11101100	{4, 40}
370	8	3/4	50.6274	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000101 00000010 00000001 11100001 11010000 11101100	{4, 40}
371	8	3/4	5	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000000 01000000 00110000 11100000	{44, 136}
372	8	3/4	5	{0, 5, 4, 3}	(0, 0)	-3.14159	00001101 00000110 00000011 00000001 10000000 11000000 01100000 10110000	{8, 20}
373	8	3/4	5	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000000 01000000 00110000 11100000	{44, 136}
374	8	3/4	5	{5, 0, 4, 3}	(1.5708, -1.5708)	3.14159	00001101 00000110 00000011 00000001 10000000 11000000 01100000 10110000	{8, 20}
375	8	3/4	48.6274	{5, 6, 5, 6}	(0, 0)	0	00001111 00000110 00000110 00000001 10000001 11100001 11100001 10011110	{4, 28}
376	8	3/4	48.6274	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	00001111 00000110 00000110 00000001 10000001 11100001 11100001 10011110	{4, 28}
377	8	3/4	46.6274	{2, 3, 2, 3}	(0, 0)	0	00001011 00000101 00000011 00000011 10000001 01000000 10110000 11111000	{16, 164}
378	8	3/4	46.6274	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000101 00000011 00000011 10000001 01000000 10110000 11111000	{16, 164}
379	8	3/4	4.74755	{0, 2, 3, 1}	(1.5708, 0)	1.23096	00001010 00001001 00000110 00000101 11000011 00110011 10101101 01011110	{4, 8}
380	8	3/4	4.74755	{0, 2, 1, 3}	(1.5708, 0)	-1.91063	00001010 00001001 00000110 00000101 11000011 00110011 10101101 01011110	{4, 8}
381	8	3/4	4.68629	{1, 2, 1, 2}	(0, 0)	0	00001001 00000111 00000111 00000011 10000000 01100010 01110100 11110000	{4, 52}
382	8	3/4	4.68629	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001001 00000111 00000111 00000011 10000000 01100010 01110100 11110000	{4, 52}
383	8	3/4	4.34315	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000001 00000001 11000010 11100100 11111000	{4, 36}
384	8	3/4	4.34315	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000001 00000001 11000010 11100100 11111000	{4, 36}
385	8	3/4	4.17157	{6, 7, 6, 7}	(0, 0)	0	00010111 00001111 00001111 10000011 01100000 11100000 11110000 11110000	{4, 32}
386	8	3/4	4.17157	{6, 7, 7, 6}	(1.5708, 0)	3.14159	00010111 00001111 00001111 10000011 01100000 11100000 11110000 11110000	{4, 32}
387	8	3/4	4	{1, 7, 2, 7}	(0, 0)	1.5708	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
388	8	3/4	4	{7, 1, 7, 2}	(0, 0)	-1.5708	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
389	8	3/4	4	{7, 1, 2, 7}	(1.5708, 0.785398)	-3.14159	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
390	8	3/4	4	{1, 7, 7, 2}	(1.5708, -0.785398)	3.14159	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
391	8	3/4	34.9706	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000001 00000001 10000001 11100000 11111100	{4, 40}
392	8	3/4	34.9706	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000001 00000001 10000001 11100000 11111100	{4, 40}
393	8	3/4	3.34315	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000001 01000001 00110000 11101100	{16, 72}
394	8	3/4	3.34315	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000001 01000001 00110000 11101100	{16, 72}
395	8	3/4	3	{1, 2, 1, 2}	(0, 0)	0	00000101 00000011 00000011 00000001 00000001 10000000 01100000 11111000	{20, 84}
396	8	3/4	3	{0, 1, 4, 3}	(1.5708, -3.14159)	1.5708	00000110 00000101 00000011 00000010 00000001 11000000 10110000 01101000	{4, 8}
397	8	3/4	3	{0, 1, 3, 4}	(1.5708, 3.14159)	-1.5708	00000110 00000101 00000011 00000010 00000001 11000000 10110000 01101000	{4, 8}
398	8	3/4	3	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000101 00000011 00000011 00000001 00000001 10000000 01100000 11111000	{20, 84}
399	8	3/4	29.6569	{6, 7, 6, 7}	(0, 0)	0	00010111 00001011 00000111 10000111 01000011 10110011 11111100 11111100	{4, 20}
400	8	3/4	29.6569	{6, 7, 7, 6}	(1.5708, 0)	3.14159	00010111 00001011 00000111 10000111 01000011 10110011 11111100 11111100	{4, 20}
401	8	3/4	29.1421	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000011 10000011 11100000 11111000 11111000	{4, 28}
402	8	3/4	29.1421	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000011 10000011 11100000 11111000 11111000	{4, 28}
403	8	3/4	27.3137	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000001 10000010 11100001 11101000 11110100	{12, 152}
404	8	3/4	27.3137	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000001 10000010 11100001 11101000 11110100	{12, 152}
405	8	3/4	25.3137	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000101 00000011 00000011 11100000 11011001 11111010	{8, 76}
406	8	3/4	25.3137	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000101 00000011 00000011 11100000 11011001 11111010	{8, 76}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
407	8	3/4	24.3137	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000000 01000000 00110001 11100010	{8, 48}
408	8	3/4	24.3137	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000000 01000000 00110001 11100010	{8, 48}
409	8	3/4	2.5	{1, 2, 1, 2}	(0, 0)	0	00001011 00000111 00000111 00000011 10000000 01100000 11110000 11110000	{4, 24}
410	8	3/4	2.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001011 00000111 00000111 00000011 10000000 01100000 11110000 11110000	{4, 24}
411	8	3/4	2.17157	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000010 00000010 10000001 01000001 00110001 11001110	{8, 32}
412	8	3/4	2.17157	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000010 00000010 10000001 01000001 00110001 11001110	{8, 32}
413	8	3/4	2	{1, 5, 2, 5}	(0, 0)	1.5708	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
414	8	3/4	2	{5, 1, 5, 2}	(0, 0)	-1.5708	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
415	8	3/4	2	{5, 1, 2, 5}	(1.5708, 0.785398)	-3.14159	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
416	8	3/4	2	{1, 5, 5, 2}	(1.5708, -0.785398)	3.14159	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
417	8	3/4	159.196	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000001 10000011 11100000 11101001 11111010	{4, 32}
418	8	3/4	159.196	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000001 10000011 11100000 11101001 11111010	{4, 32}
419	8	3/4	15.6569	{1, 2, 1, 2}	(0, 0)	0	00011111 00001111 00001111 10000010 11100001 11100001 11110000 11101100	{4, 32}
420	8	3/4	15.6569	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011111 00001111 00001111 10000010 11100001 11100001 11110000 11101100	{4, 32}
421	8	3/4	14.6569	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000001 01000001 00110001 11101110	{32, 148}
422	8	3/4	14.6569	{6, 1, 3, 4}	(0, 0)	-3.14159	00001011 00000110 00000101 00000010 10000001 01100001 11010001 10101110	{8, 32}
423	8	3/4	14.6569	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000001 01000001 00110001 11101110	{32, 148}
424	8	3/4	14.6569	{1, 6, 3, 4}	(1.5708, -1.5708)	3.14159	00001011 00000110 00000101 00000010 10000001 01100001 11010001 10101110	{8, 32}
425	8	3/4	12.6569	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000000 11110001 11111010	{12, 76}
426	8	3/4	12.6569	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000000 11110001 11111010	{12, 76}
427	8	3/4	12	{1, 2, 1, 2}	(0, 0)	0	00000101 00000011 00000011 00000010 00000001 10000000 01110000 11101000	{8, 92}
428	8	3/4	12	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000101 00000011 00000011 00000010 00000001 10000000 01110000 11101000	{8, 92}
429	8	3/4	10	{1, 2, 1, 2}	(0, 0)	0	00001001 00000111 00000111 00000001 10000000 01100000 01100000 11110000	{12, 56}
430	8	3/4	10	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001001 00000111 00000111 00000001 10000000 01100000 01100000 11110000	{12, 56}
431	8	3/4	1.8	{0, 1, 7, 6}	(1.5708, 3.14159)	0.927295	00001110 00001101 00001011 00000111 11100000 11010000 10110000 01110000	{24, 24}
432	8	3/4	1.8	{0, 1, 6, 7}	(1.5708, -3.14159)	-2.2143	00001110 00001101 00001011 00000111 11100000 11010000 10110000 01110000	{24, 24}
433	8	3/4	1.58579	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000011 10000001 01000001 00110000 11111100	{8, 32}
434	8	3/4	1.58579	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000011 10000001 01000001 00110000 11111100	{8, 32}
435	8	3/4	1.5	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000011 00000011 11000000 11111000 11111000	{4, 24}
436	8	3/4	1.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000011 11000000 11111000 11111000	{4, 24}
437	8	2/3	9.5	{1, 2, 1, 2}	(0, 0)	0	00010011 00001101 00001101 10000011 01100111 01101010 10011100 11111000	{8, 112}
438	8	2/3	9.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00010011 00001101 00001101 10000011 01100111 01101010 10011100 11111000	{8, 112}
439	8	2/3	9	{0, 6, 3, 1}	(0, 0)	0	00011011 00010111 00001100 11000101 10100010 01110000 11001001 11010010	{8, 192}
440	8	2/3	9	{0, 6, 1, 3}	(1.5708, 0)	-3.14159	00011011 00010111 00001100 11000101 10100010 01110000 11001001 11010010	{8, 192}
441	8	2/3	8	{7, 5, 7, 6}	(0, 0)	1.0472	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{18, 336}
442	8	2/3	8	{5, 7, 6, 7}	(0, 0)	-1.0472	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{18, 336}
443	8	2/3	8	{5, 7, 7, 6}	(1.5708, 0.523599)	-3.14159	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{18, 336}
444	8	2/3	8	{7, 5, 6, 7}	(1.5708, -0.523599)	3.14159	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{18, 336}
445	8	2/3	7.5	{6, 7, 6, 7}	(0, 0)	0	00001011 00000111 00000111 00000011 10000011 01100000 11111000 11111000	{16, 408}
446	8	2/3	7.5	{6, 7, 7, 6}	(1.5708, 0)	3.14159	00001011 00000111 00000111 00000011 10000011 01100000 11111000 11111000	{16, 408}
447	8	2/3	7	{0, 4, 1, 2}	(0, 0)	0	00001010 00000110 00000101 00000011 10000001 01100000 11010001 00111010	{16, 532}
448	8	2/3	7	{0, 4, 2, 1}	(1.5708, 0)	-3.14159	00001010 00000110 00000101 00000011 10000001 01100000 11010001 00111010	{16, 532}
449	8	2/3	6	{3, 2, 3, 5}	(0, 0)	2.0944	00010110 00001011 00000101 10000001 01000010 10100001 11001000 01110100	{40, 782}
450	8	2/3	6	{4, 7, 4, 6}	(0, 0)	1.0472	00011010 00010101 00001111 11000111 10100010 01110001 10111001 01110110	{5, 555}
451	8	2/3	6	{2, 3, 5, 3}	(0, 0)	-2.0944	00010110 00001011 00000101 10000001 01000010 10100001 11001000 01110100	{40, 782}
452	8	2/3	6	{7, 4, 6, 4}	(0, 0)	-1.0472	00011010 00010101 00001111 11000111 10100010 01110001 10111001 01110110	{5, 555}
453	8	2/3	6	{2, 3, 3, 5}	(1.5708, 1.0472)	-3.14159	00010110 00001011 00000101 10000001 01000010 10100001 11001000 01110100	{40, 782}
454	8	2/3	6	{7, 4, 4, 6}	(1.5708, 0.523599)	-3.14159	00011010 00010101 00001111 11000111 10100010 01110001 10111001 01110110	{5, 555}
455	8	2/3	6	{3, 2, 5, 3}	(1.5708, -1.0472)	3.14159	00010110 00001011 00000101 10000001 01000010 10100001 11001000 01110100	{40, 782}
456	8	2/3	6	{4, 7, 6, 4}	(1.5708, -0.523599)	3.14159	00011010 00010101 00001111 11000111 10100010 01110001 10111001 01110110	{5, 555}
457	8	2/3	5.5	{1, 2, 1, 2}	(0, 0)	0	00010111 00001010 00001010 10000101 01100101 10011001 11100000 10011100	{16, 428}
458	8	2/3	5.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00010111 00001010 00001010 10000101 01100101 10011001 11100000 10011100	{16, 428}
459	8	2/3	5	{7, 0, 1, 5}	(0, 0)	1.0472	00001101 00001011 00000101 00000011 11000000 10100001 01010000 11110100	{16, 876}
460	8	2/3	5	{1, 6, 5, 2}	(0, 0)	0	00000110 00000100 00000010 00000001 00000001 11000001 10100001 00011110	{104, 2404}
461	8	2/3	5	{0, 7, 5, 1}	(0, 0)	-1.0472	00001101 00001011 00000101 00000011 11000000 10100001 01010000 11110100	{16, 876}
462	8	2/3	5	{0, 7, 1, 5}	(1.5708, 0.523599)	-3.14159	00001101 00001011 00000101 00000011 11000000 10100001 01010000 11110100	{16, 876}
463	8	2/3	5	{1, 6, 2, 5}	(1.5708, 0)	3.14159	00000110 00000100 00000010 00000001 00000001 11000001 10100001 00011110	{104, 2404}
464	8	2/3	5	{7, 0, 5, 1}	(1.5708, -0.523599)	3.14159	00001101 00001011 00000101 00000011 11000000 10100001 01010000 11110100	{16, 876}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
465	8	2/3	4	{0, 4, 1, 3}	(0, 0)	1.0472	00000111 00000011 00000010 00000001 00000001 10000000 11100000 11011000	{69, 1857}
466	8	2/3	4	{4, 0, 3, 1}	(0, 0)	-1.0472	00000111 00000011 00000010 00000001 00000001 10000000 11100000 11011000	{69, 1857}
467	8	2/3	4	{4, 0, 1, 3}	(1.5708, 0.523599)	-3.14159	00000111 00000011 00000010 00000001 00000001 10000000 11100000 11011000	{69, 1857}
468	8	2/3	4	{0, 4, 3, 1}	(1.5708, -0.523599)	3.14159	00000111 00000011 00000010 00000001 00000001 10000000 11100000 11011000	{69, 1857}
469	8	2/3	3.75	{1, 2, 1, 2}	(0, 0)	0	00010111 00001011 00001011 10000100 01100101 10011000 11100001 11101010	{8, 244}
470	8	2/3	3.75	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00010111 00001011 00001011 10000100 01100101 10011000 11100001 11101010	{8, 244}
471	8	2/3	3.5	{7, 1, 7, 2}	(0, 0)	2.0944	00001111 00000101 00000011 00000001 00000001 10000111 11001000 10101000 11111000	{6, 569}
472	8	2/3	3.5	{1, 5, 2, 5}	(0, 0)	-3.14159	00010111 00001111 00000101 10000111 01000010 11110001 11011001 11110110	{11, 279}
473	8	2/3	3.5	{1, 7, 2, 7}	(0, 0)	-2.0944	00001111 00000101 00000011 00000001 10000111 11001000 10101000 11111000	{6, 569}
474	8	2/3	3.5	{1, 7, 7, 2}	(1.5708, 1.0472)	-3.14159	00001111 00000101 00000011 00000001 10000111 11001000 10101000 11111000	{6, 569}
475	8	2/3	3.5	{5, 1, 2, 5}	(1.5708, -1.5708)	3.14159	00010111 00001111 00000101 10000111 01000010 11110001 11011001 11110110	{11, 279}
476	8	2/3	3.5	{7, 1, 2, 7}	(1.5708, -1.0472)	3.14159	00001111 00000101 00000011 00000001 10000111 11001000 10101000 11111000	{6, 569}
477	8	2/3	3.33333	{1, 2, 1, 2}	(0, 0)	0	00001101 00000011 00000011 00000001 10000101 10001001 01100000 11111100	{12, 236}
478	8	2/3	3.33333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001101 00000011 00000011 00000001 10000101 10001001 01100000 11111100	{12, 236}
479	8	2/3	2.5	{1, 5, 6, 4}	(0, 0)	3.14159	00101111 00010111 10001111 01000101 10100011 11110011 11101101 11111110	{8, 88}
480	8	2/3	2.5	{5, 1, 2, 7}	(0, 0)	1.0472	00010111 00001001 00000010 10000111 01000000 10010001 10110001 11010110	{16, 84}
481	8	2/3	2.5	{1, 5, 7, 2}	(0, 0)	-1.0472	00010111 00001001 00000010 10000111 01000000 10010001 10110001 11010110	{16, 84}
482	8	2/3	2.5	{0, 4, 2, 5}	(0.588003, 3.14159)	2.24593	00001011 00001001 00000111 00000110 11000010 00110001 10111001 11100110	{2, 4}
483	8	2/3	2.5	{1, 5, 0, 2}	(0.857072, -1.5708)	-1.44547	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
484	8	2/3	2.5	{4, 0, 5, 2}	(0.588003, 0)	-2.24593	00001011 00001001 00000111 00000110 11000010 00110001 10111001 11100110	{2, 4}
485	8	2/3	2.5	{5, 1, 2, 0}	(0.857072, -1.5708)	1.44547	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
486	8	2/3	2.5	{2, 0, 5, 1}	(0.857072, 1.5708)	1.44547	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
487	8	2/3	2.5	{3, 4, 5, 6}	(0.588003, 3.14159)	-2.24593	00001111 00001111 00001110 00000011 11100101 11101000 11110001 11011010	{2, 2}
488	8	2/3	2.5	{0, 2, 1, 5}	(0.857072, 1.5708)	-1.44547	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
489	8	2/3	2.5	{4, 3, 6, 5}	(0.588003, 0)	2.24593	00001111 00001111 00001110 00000011 11100101 11101000 11110001 11011010	{2, 2}
490	8	2/3	2.5	{4, 3, 5, 6}	(1.5708, 2.0944)	2.0944	00001111 00001111 00001110 00000011 11100101 11101000 11110001 11011010	{2, 2}
491	8	2/3	2.5	{0, 2, 5, 1}	(1.0472, -2.61799)	3.14159	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
492	8	2/3	2.5	{3, 4, 6, 5}	(1.5708, -2.0944)	2.0944	00001111 00001111 00001110 00000011 11100101 11101000 11110001 11011010	{2, 2}
493	8	2/3	2.5	{2, 0, 1, 5}	(1.0472, -0.523599)	-3.14159	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
494	8	2/3	2.5	{5, 1, 0, 2}	(1.0472, 2.61799)	3.14159	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
495	8	2/3	2.5	{4, 0, 2, 5}	(1.5708, -2.0944)	-2.0944	00001011 00001001 00000111 00000110 11000010 00110001 10111001 11100110	{2, 4}
496	8	2/3	2.5	{1, 5, 2, 0}	(1.0472, 0.523599)	-3.14159	00011111 00001111 00000111 10000011 11000001 11100011 11110100 11111100	{1, 2}
497	8	2/3	2.5	{0, 4, 5, 2}	(1.5708, 2.0944)	-2.0944	00001011 00001001 00000111 00000110 11000010 00110001 10111001 11100110	{2, 4}
498	8	2/3	2.5	{1, 5, 4, 6}	(1.5708, 1.5708)	-3.14159	00101111 00010111 10001111 01000101 10100011 11110011 11101101 11111110	{8, 88}
499	8	2/3	2.5	{1, 5, 2, 7}	(1.5708, 0.523599)	-3.14159	00010111 00001001 00000010 10000111 01000000 10010001 10110001 11010110	{16, 84}
500	8	2/3	2.5	{5, 1, 7, 2}	(1.5708, -0.523599)	3.14159	00010111 00001001 00000010 10000111 01000000 10010001 10110001 11010110	{16, 84}
501	8	2/3	2.33333	{0, 3, 1, 2}	(0, 0)	2.0944	00101111 00011111 10001110 01000001 11100111 11101011 11101101 11011110	{4, 28}
502	8	2/3	2.33333	{1, 7, 6, 3}	(0, 0)	0	00101011 00010110 10001011 01000101 10100011 01010000 11101001 10111010	{8, 104}
503	8	2/3	2.33333	{3, 0, 2, 1}	(0, 0)	-2.0944	00101111 00011111 10001110 01000001 11100111 11101011 11101101 11011110	{4, 28}
504	8	2/3	2.33333	{3, 0, 1, 2}	(1.5708, 1.0472)	-3.14159	00101111 00011111 10001110 01000001 11100111 11101011 11101101 11011110	{4, 28}
505	8	2/3	2.33333	{1, 7, 3, 6}	(1.5708, 0)	3.14159	00101011 00010110 10001011 01000101 10100011 01010000 11101001 10111010	{8, 104}
506	8	2/3	2.33333	{0, 3, 2, 1}	(1.5708, -1.0472)	3.14159	00101111 00011111 10001110 01000001 11100111 11101011 11101101 11011110	{4, 28}
507	8	2/3	2.25	{0, 3, 2, 1}	(0, 0)	0	00010101 00001100 00001010 10000011 01100111 11001011 00111101 10011110	{8, 48}
508	8	2/3	2.25	{0, 3, 1, 2}	(1.5708, 0)	3.14159	00010101 00001100 00001010 10000011 01100111 11001011 00111101 10011110	{8, 48}
509	8	2/3	2	{5, 4, 1, 6}	(0.588003, -2.0944)	-2.24593	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
510	8	2/3	2	{5, 0, 6, 4}	(0.857072, -1.5708)	-1.44547	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
511	8	2/3	2	{0, 5, 4, 6}	(0.857072, -1.5708)	1.44547	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
512	8	2/3	2	{4, 5, 6, 1}	(0.588003, -1.0472)	2.24593	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
513	8	2/3	2	{4, 6, 0, 5}	(0.857072, 1.5708)	1.44547	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
514	8	2/3	2	{1, 2, 4, 3}	(0.588003, 3.14159)	-2.24593	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{4, 6}
515	8	2/3	2	{5, 4, 6, 0}	(0.857072, -2.61799)	-1.44547	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
516	8	2/3	2	{0, 3, 1, 2}	(1.5708, -2.0944)	-1.0472	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 14}
517	8	2/3	2	{1, 6, 5, 4}	(0.588003, 2.0944)	-2.24593	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
518	8	2/3	2	{6, 1, 4, 5}	(0.588003, 1.0472)	2.24593	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
519	8	2/3	2	{6, 0, 5, 4}	(0.857072, 2.61799)	-1.44547	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
520	8	2/3	2	{0, 6, 4, 5}	(0.857072, 0.523599)	1.44547	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
521	8	2/3	2	{6, 4, 5, 0}	(0.857072, 1.5708)	-1.44547	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
522	8	2/3	2	{3, 0, 2, 1}	(1.5708, 2.0944)	-1.0472	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 14}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
523	8	2/3	2	{4, 5, 0, 6}	(0.857072, -0.523599)	1.44547	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
524	8	2/3	2	{2, 1, 3, 4}	(0.588003, 0)	2.24593	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{4, 6}
525	8	2/3	2	{3, 0, 1, 2}	(1.10715, 3.14159)	2.63623	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 14}
526	8	2/3	2	{4, 5, 6, 0}	(1.28976, 2.61799)	2.24593	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
527	8	2/3	2	{2, 1, 4, 3}	(1.5708, 2.0944)	2.0944	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{4, 6}
528	8	2/3	2	{6, 4, 0, 5}	(1.0472, -2.61799)	3.14159	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
529	8	2/3	2	{6, 0, 4, 5}	(1.28976, -2.61799)	2.24593	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
530	8	2/3	2	{0, 6, 5, 4}	(1.28976, -0.523599)	-2.24593	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
531	8	2/3	2	{1, 6, 4, 5}	(1.10715, -2.0944)	2.63623	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
532	8	2/3	2	{6, 1, 5, 4}	(1.10715, -1.0472)	-2.63623	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
533	8	2/3	2	{1, 2, 3, 4}	(1.5708, -2.0944)	2.0944	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{4, 6}
534	8	2/3	2	{5, 4, 0, 6}	(1.28976, 0.523599)	-2.24593	00001111 00000111 00000011 00000001 10000110 11001001 11101001 11110110	{1, 4}
535	8	2/3	2	{0, 3, 2, 1}	(1.10715, 0)	-2.63623	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 14}
536	8	2/3	2	{4, 6, 5, 0}	(1.0472, -0.523599)	-3.14159	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
537	8	2/3	2	{0, 5, 6, 4}	(1.0472, 2.61799)	3.14159	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
538	8	2/3	2	{4, 5, 1, 6}	(1.10715, 2.0944)	2.63623	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
539	8	2/3	2	{5, 4, 6, 1}	(1.10715, 1.0472)	-2.63623	00001111 00001011 00000111 00000001 11000001 10100000 11100001 11111010	{1, 2}
540	8	2/3	2	{5, 0, 4, 6}	(1.0472, 0.523599)	-3.14159	00001111 00000111 00000110 00000011 10000001 11100001 11110001 11011110	{1, 3}
541	8	2/3	16	{1, 2, 1, 2}	(0, 0)	0	00001011 00000101 00000101 00000011 10000001 01100010 10010100 11111000	{12, 492}
542	8	2/3	16	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001011 00000101 00000101 00000011 10000001 01100010 10010100 11111000	{12, 492}
543	8	2/3	12	{1, 2, 1, 2}	(0, 0)	0	00001010 00000111 00000111 00000001 10000001 01100001 11100000 01111100	{16, 1100}
544	8	2/3	12	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001010 00000111 00000111 00000001 10000001 01100001 11100000 01111100	{16, 1100}
545	8	2/3	10	{1, 5, 2, 5}	(0, 0)	-3.14159	00011101 00001111 00000111 10000011 11000001 11100001 01110000 11111100	{4, 118}
546	8	2/3	10	{5, 1, 2, 5}	(1.5708, -1.5708)	3.14159	00011101 00001111 00000111 10000011 11000001 11100001 01110000 11111100	{4, 118}
547	8	2/3	1.75	{1, 2, 1, 2}	(0, 0)	0	00001111 00000101 00000101 00000011 10000010 11100011 10011101 11110110	{48, 776}
548	8	2/3	1.75	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000101 00000101 00000011 10000010 11100011 10011101 11110110	{48, 776}
549	8	2/3	1.625	{6, 4, 5, 7}	(1.5708, -2.0944)	-1.0472	00010111 00001111 00001011 10000111 01100110 11011001 11111001 11110110	{4, 4}
550	8	2/3	1.625	{4, 6, 7, 5}	(1.5708, 2.0944)	-1.0472	00010111 00001111 00001011 10000111 01100110 11011001 11111001 11110110	{4, 4}
551	8	2/3	1.625	{4, 6, 5, 7}	(1.10715, 3.14159)	2.63623	00010111 00001111 00001011 10000111 01100110 11011001 11111001 11110110	{4, 4}
552	8	2/3	1.625	{6, 4, 7, 5}	(1.10715, 0)	-2.63623	00010111 00001111 00001011 10000111 01100110 11011001 11111001 11110110	{4, 4}
553	8	2/3	1.57143	{0, 1, 5, 4}	(1.5708, -3.14159)	1.42745	00011010 00010110 00001111 11000011 10100001 01100001 11110000 00111100	{4, 4}
554	8	2/3	1.57143	{0, 1, 4, 5}	(1.5708, 3.14159)	-1.71414	00011010 00010110 00001111 11000011 10100001 01100001 11110000 00111100	{4, 4}
555	8	2/3	1.33333	{1, 3, 2, 3}	(0, 0)	2.0944	00010001 00001111 00001111 10000000 01100111 01101011 01101101 11101110	{8, 38}
556	8	2/3	1.33333	{3, 1, 3, 2}	(0, 0)	-2.0944	00010001 00001111 00001111 10000000 01100111 01101011 01101101 11101110	{8, 38}
557	8	2/3	1.33333	{3, 1, 2, 3}	(1.5708, 1.0472)	-3.14159	00010001 00001111 00001111 10000000 01100111 01101011 01101101 11101110	{8, 38}
558	8	2/3	1.33333	{1, 3, 3, 2}	(1.5708, -1.0472)	3.14159	00010001 00001111 00001111 10000000 01100111 01101011 01101101 11101110	{8, 38}
559	8	2/3	1.25	{1, 2, 1, 2}	(0, 0)	0	00011011 00000100 00000100 10001011 10010011 01100001 10011001 10011110	{16, 88}
560	8	2/3	1.25	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011011 00000100 00000100 10001011 10010011 01100001 10011001 10011110	{16, 88}
561	8	2/3	1	{3, 2, 3, 6}	(0, 0)	2.0944	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{6, 54}
562	8	2/3	1	{2, 3, 6, 3}	(0, 0)	-2.0944	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{6, 54}
563	8	2/3	1	{2, 3, 3, 6}	(1.5708, 1.0472)	-3.14159	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{6, 54}
564	8	2/3	1	{3, 2, 6, 3}	(1.5708, -1.0472)	3.14159	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{6, 54}
565	8	2/3	0.857143	{2, 3, 2, 3}	(0, 0)	0	00001011 00000101 00000001 00000001 10000010 01000000 10001000 11110000	{8, 120}
566	8	2/3	0.857143	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000101 00000001 00000001 10000010 01000000 10001000 11110000	{8, 120}
567	8	2/3	0.833333	{1, 2, 1, 2}	(0, 0)	0	00010111 00001000 00001000 10000111 01100111 10011011 10011101 10011110	{8, 48}
568	8	2/3	0.833333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00010111 00001000 00001000 10000111 01100111 10011011 10011101 10011110	{8, 48}
569	8	3/5	99.1935	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 12}
570	8	3/5	99.1935	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 12}
571	8	3/5	8.23607	{2, 6, 5, 3}	(0, 0)	0	00001001 00000110 00000100 00000010 10000001 01100001 01010001 10001110	{8, 88}
572	8	3/5	8.23607	{2, 6, 3, 5}	(1.5708, 0)	-3.14159	00001001 00000110 00000100 00000010 10000001 01100001 01010001 10001110	{8, 88}
573	8	3/5	75.7771	{0, 1, 0, 1}	(0, 0)	0	00001011 00001011 00000110 00000100 11000000 00110001 11100001 11000110	{4, 56}
574	8	3/5	75.7771	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001011 00001011 00000110 00000100 11000000 00110001 11100001 11000110	{4, 56}
575	8	3/5	5.78885	{2, 3, 2, 3}	(0, 0)	0	00001110 00001000 00000111 00000111 11000011 10110001 10111000 00111100	{4, 36}
576	8	3/5	5.78885	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001110 00001000 00000111 00000111 11000011 10110001 10111000 00111100	{4, 36}
577	8	3/5	37.8885	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{4, 56}
578	8	3/5	37.8885	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{4, 56}
579	8	3/5	3.61803	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000001 11000001 10110000 10111100	{8, 104}
580	8	3/5	3.61803	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000001 11000001 10110000 10111100	{8, 104}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
581	8	3/5	3.21603	{2, 3, 2, 3}	(0, 0)	0	00001111 00001010 00000111 00000111 11000001 10110000 11110001 10111010	{4, 44}
582	8	3/5	3.21603	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001111 00001010 00000111 00000111 11000001 10110000 11110001 10111010	{4, 44}
583	8	3/5	3	{2, 5, 4, 3}	(0, 0)	0	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 16}
584	8	3/5	3	{2, 5, 3, 4}	(1.5708, 0)	-3.14159	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 16}
585	8	3/5	15.1554	{2, 3, 2, 3}	(0, 0)	0	00001111 00001011 00000111 00000111 11000110 10111001 11111000 11110100	{4, 28}
586	8	3/5	15.1554	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001111 00001011 00000111 00000111 11000110 10111001 11111000 11110100	{4, 28}
587	8	3/5	14.4721	{2, 3, 2, 3}	(0, 0)	0	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{4, 36}
588	8	3/5	14.4721	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{4, 36}
589	8	3/5	11	{0, 4, 3, 1}	(0, 0)	0	00010011 00001011 00000101 10000011 01000011 00100001 11011000 11111100	{8, 48}
590	8	3/5	11	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	00010011 00001011 00000101 10000011 01000011 00100001 11011000 11111100	{8, 48}
591	8	3/5	1.60802	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000001 11110000 11111100	{4, 32}
592	8	3/5	1.60802	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000001 11110000 11111100	{4, 32}
593	8	1/2	7	{2, 3, 2, 3}	(0, 0)	0	00001111 00000111 00000011 00000011 10000001 11000010 11110100 11111000	{332, 3072}
594	8	1/2	7	{0, 2, 1, 2}	(0, 0)	-3.14159	00001101 00001001 00000111 00000011 11000011 10100001 00111000 11111100	{50, 368}
595	8	1/2	7	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001111 00000111 00000011 00000011 10000001 11000010 11110100 11111000	{332, 3072}
596	8	1/2	7	{2, 0, 1, 2}	(1.5708, -1.5708)	3.14159	00001101 00001001 00000111 00000011 11000011 10100001 00111000 11111100	{50, 368}
597	8	1/2	6	{2, 3, 2, 3}	(0, 0)	0	00000110 00000101 00000011 00000011 00000001 11000000 10110000 01111000	{260, 3332}
598	8	1/2	6	{7, 1, 5, 1}	(0, 0)	-3.14159	00001101 00001010 00000111 00000010 11000001 10100000 01110000 10101000	{60, 603}
599	8	1/2	6	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000110 00000101 00000011 00000011 00000001 11000000 10110000 01111000	{260, 3332}
600	8	1/2	6	{1, 7, 5, 1}	(1.5708, -1.5708)	3.14159	00001101 00001010 00000111 00000010 11000001 10100000 01110000 10101000	{60, 603}
601	8	1/2	5	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000001 01000001 00110000 11101100	{846, 8156}
602	8	1/2	5	{3, 0, 2, 5}	(0, 0)	-3.14159	00001110 00000111 00000001 00000001 10000011 11000001 11001000 01111100	{284, 2001}
603	8	1/2	5	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000001 01000001 00110000 11101100	{846, 8156}
604	8	1/2	5	{0, 3, 2, 5}	(1.5708, -1.5708)	3.14159	00001110 00000111 00000001 00000001 10000011 11000001 11001000 01111100	{284, 2001}
605	8	1/2	4.5	{1, 5, 6, 4}	(0, 0)	1.5708	00001011 00001001 00000111 00000111 11000110 00111001 10111000 11110100	{16, 44}
606	8	1/2	4.5	{5, 1, 4, 6}	(0, 0)	-1.5708	00001011 00001001 00000111 00000111 11000110 00111001 10111000 11110100	{16, 44}
607	8	1/2	4.5	{5, 1, 6, 4}	(1.5708, 0.785398)	-3.14159	00001011 00001001 00000111 00000111 11000110 00111001 10111000 11110100	{16, 44}
608	8	1/2	4.5	{1, 5, 4, 6}	(1.5708, -0.785398)	3.14159	00001011 00001001 00000111 00000111 11000110 00111001 10111000 11110100	{16, 44}
609	8	1/2	4	{0, 3, 1, 2}	(0, 0)	0	00000101 00000100 00000011 00000010 00000001 11000001 00110001 10101110	{1710, 14142}
610	8	1/2	4	{0, 3, 4, 2}	(0, 0)	-3.14159	00000111 00000110 00000100 00000010 00000001 11100001 11010001 10001110	{453, 3423}
611	8	1/2	4	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	00000101 00000100 00000011 00000010 00000001 11000001 00110001 10101110	{1710, 14142}
612	8	1/2	4	{3, 0, 4, 2}	(1.5708, -1.5708)	3.14159	00000111 00000110 00000100 00000010 00000001 11100001 11010001 10001110	{453, 3423}
613	8	1/2	3.5	{0, 6, 4, 5}	(0, 0)	1.5708	00001110 00000111 00000111 00000001 10000001 11100001 11100000 01111100	{16, 88}
614	8	1/2	3.5	{6, 0, 5, 4}	(0, 0)	-1.5708	00001110 00000111 00000111 00000001 10000001 11100001 11100000 01111100	{16, 88}
615	8	1/2	3.5	{6, 0, 4, 5}	(1.5708, 0.785398)	-3.14159	00001110 00000111 00000111 00000001 10000001 11100001 11100000 01111100	{16, 88}
616	8	1/2	3.5	{0, 6, 5, 4}	(1.5708, -0.785398)	3.14159	00001110 00000111 00000111 00000001 10000001 11100001 11100000 01111100	{16, 88}
617	8	1/2	2.8	{1, 4, 2, 3}	(1.5708, 0)	-0.927295	00011011 00001111 00000110 10000001 11000011 01100011 11101101 11011110	{8, 8}
618	8	1/2	2.8	{1, 4, 3, 2}	(1.5708, 3.14159)	-2.2143	00011011 00001111 00000110 10000001 11000011 01100011 11101101 11011110	{8, 8}
619	8	1/2	2.6	{1, 3, 4, 2}	(1.5708, 0)	-0.927295	00011111 00010110 00001101 11000010 10100001 11100011 11010101 10101110	{4, 8}
620	8	1/2	2.6	{1, 3, 2, 4}	(1.5708, -3.14159)	-2.2143	00011111 00010110 00001101 11000010 10100001 11100011 11010101 10101110	{4, 8}
621	8	1/2	2.5	{6, 0, 2, 1}	(0, 0)	1.5708	00000110 00000110 00000010 00000001 00000001 11000001 11100000 00011100	{154, 939}
622	8	1/2	2.5	{0, 6, 1, 2}	(0, 0)	-1.5708	00000110 00000110 00000010 00000001 00000001 11000001 11100000 00011100	{154, 939}
623	8	1/2	2.5	{0, 6, 2, 1}	(1.5708, 0.785398)	-3.14159	00000110 00000110 00000010 00000001 00000001 11000001 11100000 00011100	{154, 939}
624	8	1/2	2.5	{6, 0, 1, 2}	(1.5708, -0.785398)	3.14159	00000110 00000110 00000010 00000001 00000001 11000001 11100000 00011100	{154, 939}
625	8	1/2	2.33333	{0, 2, 4, 1}	(0, 0)	1.5708	00001001 00000111 00000110 00000001 10000001 01100001 01100000 11011100	{28, 234}
626	8	1/2	2.33333	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000011 00000011 11000001 11111000 11111100	{48, 532}
627	8	1/2	2.33333	{0, 6, 1, 6}	(0, 0)	-3.14159	00000101 00000101 00000011 00000011 00000011 11000011 00111100 11111100	{16, 48}
628	8	1/2	2.33333	{2, 0, 1, 4}	(0, 0)	-1.5708	00001001 00000111 00000110 00000001 10000001 01100001 01100000 11011100	{28, 234}
629	8	1/2	2.33333	{2, 0, 4, 1}	(1.5708, 0.785398)	-3.14159	00001001 00000111 00000110 00000001 10000001 01100001 01100000 11011100	{28, 234}
630	8	1/2	2.33333	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000011 11000001 11111000 11111100	{48, 532}
631	8	1/2	2.33333	{6, 0, 1, 6}	(1.5708, -1.5708)	3.14159	00000101 00000101 00000011 00000011 00000011 11000011 00111100 11111100	{16, 48}
632	8	1/2	2.33333	{0, 2, 1, 4}	(1.5708, -0.785398)	3.14159	00001001 00000111 00000110 00000001 10000001 01100001 01100000 11011100	{28, 234}
633	8	1/2	2	{3, 6, 5, 4}	(0.463648, 3.14159)	3.14159	00101111 00011011 10001110 01000111 11100101 10111000 11110001 11011010	{2, 4}
634	8	1/2	2	{6, 1, 5, 2}	(1.5708, -1.5708)	-0.927295	00010111 00001111 00001001 10000110 01100011 11010001 11011001 11101110	{2, 4}
635	8	1/2	2	{1, 2, 4, 5}	(1.5708, 0)	-0.927295	00010011 00001011 00000111 10000011 01000001 00100001 11110001 11111110	{16, 16}
636	8	1/2	2	{1, 6, 2, 5}	(1.5708, 1.5708)	-0.927295	00010111 00001111 00001001 10000110 01100011 11010001 11011001 11101110	{2, 4}
637	8	1/2	2	{6, 3, 4, 5}	(0.463648, 0)	3.14159	00101111 00011011 10001110 01000111 11100101 10111000 11110001 11011010	{2, 4}
638	8	1/2	2	{1, 5, 2, 4}	(1.5708, -3.14159)	-0.927295	00010011 00001111 00001101 10000010 01100011 01100001 11011001 11101110	{4, 4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
639	8	1/2	2	{1, 6, 5, 2}	(1.10715, 3.14159)	3.14159	00010111 00001111 00001001 10000110 01100011 11010001 11011001 11101110	{2, 4}
640	8	1/2	2	{6, 3, 5, 4}	(1.5708, -1.5708)	-2.2143	00101111 00011011 10001110 01000111 11100101 10111000 11110001 11011010	{2, 4}
641	8	1/2	2	{1, 5, 4, 2}	(1.5708, 0)	-2.2143	00010011 00001111 00001101 10000010 01100011 01100001 11011001 11101110	{4, 4}
642	8	1/2	2	{3, 6, 4, 5}	(1.5708, 1.5708)	-2.2143	00101111 00011011 10001110 01000111 11100101 10111000 11110001 11011010	{2, 4}
643	8	1/2	2	{6, 1, 2, 5}	(1.10715, 0)	3.14159	00010111 00001111 00001001 10000110 01100011 11010001 11011001 11101110	{2, 4}
644	8	1/2	2	{1, 2, 5, 4}	(1.5708, -3.14159)	-2.2143	00010011 00001011 00000111 10000011 01000001 00100001 11110001 11111110	{16, 16}
645	8	1/2	11	{1, 2, 1, 2}	(0, 0)	0	00001110 00000101 00000101 00000011 10000011 11100010 10011101 01111010	{48, 668}
646	8	1/2	11	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001110 00000101 00000101 00000011 10000011 11100010 10011101 01111010	{48, 668}
647	8	1/2	10	{2, 3, 2, 3}	(0, 0)	0	00001111 00001101 00000011 00000011 11000010 11000001 10111000 11110100	{64, 840}
648	8	1/2	10	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001111 00001101 00000011 00000011 11000010 11000001 10111000 11110100	{64, 840}
649	8	1/2	1.85714	{0, 1, 0, 1}	(0, 0)	0	00001011 00001011 00000111 00000111 11000110 00111001 11111000 11110100	{16, 176}
650	8	1/2	1.85714	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001011 00001011 00000111 00000111 11000110 00111001 11111000 11110100	{16, 176}
651	8	1/2	1.8	{1, 2, 4, 3}	(1.5708, 0)	-0.927295	00011111 00010110 00001110 11000001 10100001 11100011 11100101 10011110	{4, 8}
652	8	1/2	1.8	{1, 2, 3, 4}	(1.5708, -3.14159)	-2.2143	00011111 00010110 00001110 11000001 10100001 11100011 11100101 10011110	{4, 8}
653	8	1/2	1.75	{1, 4, 2, 3}	(0, 0)	0	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{168, 2836}
654	8	1/2	1.75	{3, 1, 2, 1}	(0, 0)	-3.14159	00001110 00000101 00000011 00000011 10000101 11001000 10110000 01111000	{56, 630}
655	8	1/2	1.75	{1, 4, 3, 2}	(1.5708, 0)	-3.14159	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{168, 2836}
656	8	1/2	1.75	{1, 3, 2, 1}	(1.5708, -1.5708)	3.14159	00001110 00000101 00000011 00000011 10000101 11001000 10110000 01111000	{56, 630}
657	8	1/2	1.66667	{1, 3, 5, 2}	(0, 0)	1.5708	00001101 00000100 00000011 00000011 10000000 11000000 00110000 10110000	{54, 237}
658	8	1/2	1.66667	{3, 1, 2, 5}	(0, 0)	-1.5708	00001101 00000100 00000011 00000011 10000000 11000000 00110000 10110000	{54, 237}
659	8	1/2	1.66667	{3, 1, 5, 2}	(1.5708, 0.785398)	-3.14159	00001101 00000100 00000011 00000011 10000000 11000000 00110000 10110000	{54, 237}
660	8	1/2	1.66667	{1, 3, 2, 5}	(1.5708, -0.785398)	3.14159	00001101 00000100 00000011 00000011 10000000 11000000 00110000 10110000	{54, 237}
661	8	1/2	1.6	{0, 3, 1, 2}	(0, 0)	0	00001011 00001011 00000111 00000111 11000000 00110000 11110000 11110000	{40, 168}
662	8	1/2	1.6	{2, 4, 3, 4}	(0, 0)	-3.14159	00001001 00001001 00000111 00000111 11000010 00110000 00111001 11110010	{16, 47}
663	8	1/2	1.6	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	00001011 00001011 00000111 00000111 11000000 00110000 11110000 11110000	{40, 168}
664	8	1/2	1.6	{4, 2, 3, 4}	(1.5708, -1.5708)	3.14159	00001001 00001001 00000111 00000111 11000010 00110000 00111001 11110010	{16, 47}
665	8	1/2	1.4	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000110 00000001 00000001 11100001 11100000 11011100	{8, 88}
666	8	1/2	1.4	{6, 0, 5, 0}	(0, 0)	-3.14159	00001111 00001001 00000110 00000110 11000000 10110001 10110001 11000110	{10, 34}
667	8	1/2	1.4	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000110 00000001 00000001 11100001 11100000 11011100	{8, 88}
668	8	1/2	1.4	{0, 6, 5, 0}	(1.5708, -1.5708)	3.14159	00001111 00001001 00000110 00000110 11000000 10110001 10110001 11000110	{10, 34}
669	8	1/2	1.33333	{1, 2, 1, 2}	(0, 0)	0	00000011 00000010 00000010 00000001 00000001 00000001 11100000 10011100	{48, 324}
670	8	1/2	1.33333	{7, 0, 6, 0}	(0, 0)	-3.14159	00000111 00000100 00000011 00000011 00000011 11000000 10111000 10111000	{8, 32}
671	8	1/2	1.33333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000011 00000010 00000010 00000001 00000001 00000001 11100000 10011100	{48, 324}
672	8	1/2	1.33333	{0, 7, 6, 0}	(1.5708, -1.5708)	3.14159	00000111 00000100 00000011 00000011 00000011 11000000 10111000 10111000	{8, 32}
673	8	1/2	1.25	{0, 7, 5, 6}	(0, 0)	1.5708	00000111 00000011 00000011 00000011 00000011 10000000 11111000 11111000	{4, 4}
674	8	1/2	1.25	{7, 0, 6, 5}	(0, 0)	-1.5708	00000111 00000011 00000011 00000011 00000011 10000000 11111000 11111000	{4, 4}
675	8	1/2	1.25	{7, 0, 5, 6}	(1.5708, 0.785398)	-3.14159	00000111 00000011 00000011 00000011 00000011 10000000 11111000 11111000	{4, 4}
676	8	1/2	1.25	{0, 7, 6, 5}	(1.5708, -0.785398)	3.14159	00000111 00000011 00000011 00000011 00000011 10000000 11111000 11111000	{4, 4}
677	8	1/2	1.16667	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000100 00000011 00000011 11100001 11011000 11011100	{8, 40}
678	8	1/2	1.16667	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000100 00000011 00000011 11100001 11011000 11011100	{8, 40}
679	8	1/2	0.857143	{3, 4, 3, 4}	(0, 0)	0	00000111 00000111 00000110 00000001 00000001 11100001 11100000 11011100	{8, 1148}
680	8	1/2	0.857143	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	00000111 00000111 00000110 00000001 00000001 11100001 11100000 11011100	{8, 1148}
681	8	1/2	0.714286	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000011 11000001 11111000 11111100	{168, 1124}
682	8	1/2	0.714286	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000011 11000001 11111000 11111100	{168, 1124}
683	8	1/2	0.625	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000011 11000001 10111000 10111100	{88, 632}
684	8	1/2	0.625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000011 11000001 10111000 10111100	{88, 632}
685	8	1/2	0.6	{3, 4, 3, 4}	(0, 0)	0	00000011 00000010 00000010 00000001 00000001 00000001 11100000 10011100	{168, 780}
686	8	1/2	0.6	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	00000011 00000010 00000010 00000001 00000001 00000001 11100000 10011100	{168, 780}
687	8	1/2	0.428571	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000011 00000011 10000001 11111000 11111100	{48, 384}
688	8	1/2	0.428571	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000011 00000011 10000001 11111000 11111100	{48, 384}
689	8	1/2	0.4	{2, 3, 2, 3}	(0, 0)	0	00000011 00000010 00000001 00000001 00000001 00000001 11000000 10111100	{96, 480}
690	8	1/2	0.4	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000011 00000010 00000001 00000001 00000001 00000001 11000000 10111100	{96, 480}
691	8	1/2	0.2	{0, 1, 0, 1}	(0, 0)	0	00000001 00000001 00000001 00000001 00000001 00000001 00000001 11111110	{84, 252}
692	8	1/2	0.2	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000001 00000001 00000001 00000001 00000001 00000001 00000001 11111110	{84, 252}
693	8	2/5	8.2918	{6, 7, 6, 7}	(0, 0)	0	00000111 00000111 00000111 00000011 00000011 11100011 11111100 11111100	{4, 40}
694	8	2/5	8.2918	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	00000111 00000111 00000111 00000011 00000011 11100011 11111100 11111100	{4, 40}
695	8	2/5	8.23607	{0, 5, 4, 1}	(0, 0)	0	00001011 00000111 00000001 00000001 10000011 01000011 11001101 11111110	{8, 80}
696	8	2/5	8.23607	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001011 00000111 00000001 00000001 10000011 01000011 11001101 11111110	{8, 80}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
697	8	2/5	6.54509	{2, 3, 2, 3}	(0, 0)	0	00001101 00000111 00000001 00000001 10000010 11000000 01001000 11110000	{8, 64}
698	8	2/5	6.54509	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001101 00000111 00000001 00000001 10000010 11000000 01001000 11110000	{8, 64}
699	8	2/5	56.8328	{5, 6, 5, 6}	(0, 0)	0	00001111 00000111 00000111 00000111 10000111 11111001 11111001 11111110	{4, 16}
700	8	2/5	56.8328	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000111 10000111 11111001 11111001 11111110	{4, 16}
701	8	2/5	4.34164	{2, 3, 2, 3}	(0, 0)	0	00001011 00000111 00000101 00000101 10000011 01110000 11001000 11111000	{4, 32}
702	8	2/5	4.34164	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000111 00000101 00000101 10000011 01110000 11001000 11111000	{4, 32}
703	8	2/5	3.81966	{5, 6, 5, 6}	(0, 0)	0	00011111 00001111 00000110 10001111 11010111 11111001 11111001 11011110	{4, 108}
704	8	2/5	3.81966	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	00011111 00001111 00000110 10001111 11010111 11111001 11111001 11011110	{4, 108}
705	8	2/5	3.76393	{1, 6, 5, 2}	(0, 0)	0	00000111 00000100 00000010 00000001 00000001 11000001 10100001 10011110	{8, 56}
706	8	2/5	3.76393	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	00000111 00000100 00000010 00000001 00000001 11000001 10100001 10011110	{8, 56}
707	8	2/5	3.16718	{5, 6, 5, 6}	(0, 0)	0	00011111 00011111 00000110 11001111 11010111 11111001 11111001 11011110	{4, 48}
708	8	2/5	3.16718	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	00011111 00011111 00000110 11001111 11010111 11111001 11111001 11011110	{4, 48}
709	8	2/5	3	{2, 5, 4, 3}	(0, 0)	0	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 8}
710	8	2/5	3	{2, 5, 3, 4}	(1.5708, 0)	3.14159	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 8}
711	8	2/5	26.1803	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00000100 00000010 11000001 11100001 11010001 11001110	{8, 84}
712	8	2/5	26.1803	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001111 00001111 00000100 00000010 11000001 11100001 11010001 11001110	{8, 84}
713	8	2/5	21.7082	{1, 2, 1, 2}	(0, 0)	0	00001011 00000101 00000101 00000010 10000011 01100000 10011000 11101000	{12, 116}
714	8	2/5	21.7082	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001011 00000101 00000101 00000010 10000011 01100000 10011000 11101000	{12, 116}
715	8	2/5	2.5	{2, 3, 2, 3}	(0, 0)	0	00001100 00000101 00000010 00000010 10000001 11000000 00110001 01001010	{8, 64}
716	8	2/5	2.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001100 00000101 00000010 00000010 10000001 11000000 00110001 01001010	{8, 64}
717	8	2/5	2.11146	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{12, 232}
718	8	2/5	2.11146	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{12, 232}
719	8	2/5	2.07295	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000010 00000001 11000011 11110101 11101110	{8, 88}
720	8	2/5	2.07295	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000010 00000001 11000011 11110101 11101110	{8, 88}
721	8	2/5	14.4721	{2, 3, 2, 3}	(0, 0)	0	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{24, 432}
722	8	2/5	14.4721	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{24, 432}
723	8	2/5	10	{1, 2, 1, 2}	(0, 0)	0	00000110 00000101 00000101 00000011 00000011 11100001 10011001 01111110	{24, 320}
724	8	2/5	10	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000110 00000101 00000101 00000011 00000011 11100001 10011001 01111110	{24, 320}
725	8	2/5	1.45898	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00001011 00000111 11100001 11010001 11110001 11111110	{4, 60}
726	8	2/5	1.45898	{0, 1, 1, 0}	(1.5708, 0)	3.14159	00001111 00001111 00001011 00000111 11100001 11010001 11110001 11111110	{4, 60}
727	8	2/5	1.20976	{2, 3, 2, 3}	(0, 0)	0	00001011 00000111 00000101 00000101 10000011 01110010 11001101 11111010	{4, 32}
728	8	2/5	1.20976	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000111 00000101 00000101 10000011 01110010 11001101 11111010	{4, 32}
729	8	2/5	1.10557	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 36}
730	8	2/5	1.10557	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 36}
731	8	2/5	0.954915	{2, 3, 2, 3}	(0, 0)	0	00001101 00000110 00000001 00000001 10000011 11000000 01001000 10111000	{8, 64}
732	8	2/5	0.954915	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001101 00000110 00000001 00000001 10000011 11000000 01001000 10111000	{8, 64}
733	8	2/5	0.806504	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000101 10000011 11110001 11101001 11111110	{4, 44}
734	8	2/5	0.806504	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000101 10000011 11110001 11101001 11111110	{4, 44}
735	8	1/3	8.5	{0, 4, 3, 1}	(0, 0)	0	00010111 00001111 00000011 10000111 01000111 11011000 11111001 11111010	{8, 266}
736	8	1/3	8.5	{0, 4, 1, 3}	(1.5708, 0)	3.14159	00010111 00001111 00000011 10000111 01000111 11011000 11111001 11111010	{8, 266}
737	8	1/3	8	{4, 7, 6, 7}	(0, 0)	2.0944	00001111 00000111 00000011 00000011 10000001 11000000 11110000 11111000	{8, 237}
738	8	1/3	8	{5, 7, 6, 7}	(0, 0)	1.0472	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{6, 137}
739	8	1/3	8	{7, 4, 7, 6}	(0, 0)	-2.0944	00001111 00000111 00000011 00000011 10000001 11000000 11110000 11111000	{8, 237}
740	8	1/3	8	{7, 5, 7, 6}	(0, 0)	-1.0472	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{6, 137}
741	8	1/3	8	{7, 4, 6, 7}	(1.5708, 1.0472)	-3.14159	00001111 00000111 00000011 00000011 10000001 11000000 11110000 11111000	{8, 237}
742	8	1/3	8	{7, 5, 6, 7}	(1.5708, 0.523599)	-3.14159	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{6, 137}
743	8	1/3	8	{4, 7, 7, 6}	(1.5708, -1.0472)	3.14159	00001111 00000111 00000011 00000011 10000001 11000000 11110000 11111000	{8, 237}
744	8	1/3	8	{5, 7, 7, 6}	(1.5708, -0.523599)	3.14159	00001101 00001011 00000101 00000011 11000000 10100000 01010000 11110000	{6, 137}
745	8	1/3	7	{0, 7, 4, 6}	(0, 0)	2.0944	00001011 00000110 00000001 00000001 10000011 01000001 11001001 10111110	{4, 56}
746	8	1/3	7	{7, 0, 6, 4}	(0, 0)	-2.0944	00001011 00000110 00000001 00000001 10000011 01000001 11001001 10111110	{4, 56}
747	8	1/3	7	{7, 0, 4, 6}	(1.5708, 1.0472)	-3.14159	00001011 00000110 00000001 00000001 10000011 01000001 11001001 10111110	{4, 56}
748	8	1/3	7	{0, 7, 6, 4}	(1.5708, -1.0472)	3.14159	00001011 00000110 00000001 00000001 10000011 01000001 11001001 10111110	{4, 56}
749	8	1/3	6.5	{2, 3, 2, 3}	(0, 0)	0	00001100 00001011 00000111 00000111 11000011 10110000 01111001 01111010	{8, 148}
750	8	1/3	6.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001100 00001011 00000111 00000111 11000011 10110000 01111001 01111010	{8, 148}
751	8	1/3	6	{0, 1, 0, 1}	(0, 0)	0	00000011 00000011 00000010 00000001 00000001 00000001 11100001 11011110	{24, 1324}
752	8	1/3	6	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000011 00000011 00000010 00000001 00000001 00000001 11100001 11011110	{24, 1324}
753	8	1/3	5	{0, 6, 5, 2}	(0, 0)	3.14159	00001010 00000110 00000010 00000001 10000001 01000001 11100001 00011110	{32, 256}
754	8	1/3	5	{0, 2, 7, 1}	(0, 0)	2.0944	00001101 00001001 00000110 00000011 11000010 10100001 00111000 11010100	{4, 232}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
755	8	1/3	5	{0, 6, 3, 5}	(0, 0)	1.0472	00001100 00001010 00000101 00000011 11000001 10100000 01010000 00111000	{16, 224}
756	8	1/3	5	{1, 6, 5, 2}	(0, 0)	0	00000110 00000100 00000010 00000001 00000001 11000001 10100001 00011110	{56, 768}
757	8	1/3	5	{2, 0, 1, 7}	(0, 0)	-2.0944	00001101 00001001 00000110 00000011 11000010 10100001 00111000 11010100	{4, 232}
758	8	1/3	5	{6, 0, 5, 3}	(0, 0)	-1.0472	00001100 00001010 00000101 00000011 11000001 10100000 01010000 00111000	{16, 224}
759	8	1/3	5	{0, 6, 2, 5}	(1.5708, 1.5708)	-3.14159	00001010 00000110 00000010 00000001 10000001 01000001 11100001 00011110	{32, 256}
760	8	1/3	5	{2, 0, 7, 1}	(1.5708, 1.0472)	-3.14159	00001101 00001001 00000110 00000011 11000010 10100001 00111000 11010100	{4, 232}
761	8	1/3	5	{6, 0, 3, 5}	(1.5708, 0.523599)	-3.14159	00001100 00001010 00000101 00000011 11000001 10100000 01010000 00111000	{16, 224}
762	8	1/3	5	{1, 6, 2, 5}	(1.5708, 0)	3.14159	00000110 00000100 00000010 00000001 00000001 11000001 10100001 00011110	{56, 768}
763	8	1/3	5	{0, 2, 1, 7}	(1.5708, -1.0472)	3.14159	00001101 00001001 00000110 00000011 11000010 10100001 00111000 11010100	{4, 232}
764	8	1/3	5	{0, 6, 5, 3}	(1.5708, -0.523599)	3.14159	00001100 00001010 00000101 00000011 11000001 10100000 01010000 00111000	{16, 224}
765	8	1/3	4.5	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000001 00000001 00000001 11000011 11000101 11111110	{20, 1018}
766	8	1/3	4.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000001 00000001 00000001 11000011 11000101 11111110	{20, 1018}
767	8	1/3	4	{0, 5, 2, 5}	(0, 0)	-3.14159	00001010 00000101 00000011 00000010 10000001 01000000 10110000 01101000	{12, 204}
768	8	1/3	4	{5, 0, 2, 5}	(1.5708, -1.5708)	3.14159	00001010 00000101 00000011 00000010 10000001 01000000 10110000 01101000	{12, 204}
769	8	1/3	34	{1, 2, 1, 2}	(0, 0)	0	00010110 00001011 00001011 10000101 01100101 10011001 11100000 01111100	{4, 108}
770	8	1/3	34	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00010110 00001011 00001011 10000101 01100101 10011001 11100000 01111100	{4, 108}
771	8	1/3	3.77778	{1, 2, 1, 2}	(0, 0)	0	00010111 00001011 00001011 10000110 01100101 10011001 11110000 11101100	{4, 96}
772	8	1/3	3.77778	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00010111 00001011 00001011 10000110 01100101 10011001 11110000 11101100	{4, 96}
773	8	1/3	3.5	{1, 7, 6, 3}	(0, 0)	3.14159	00001111 00000101 00000011 00000001 10000111 11001000 10101000 11111000	{16, 136}
774	8	1/3	3.5	{2, 6, 5, 3}	(0, 0)	0	00001111 00000110 00000100 00000010 10000001 11100001 11010001 10001110	{40, 344}
775	8	1/3	3.5	{1, 7, 3, 6}	(1.5708, 1.5708)	-3.14159	00001111 00000101 00000011 00000001 10000111 11001000 10101000 11111000	{16, 136}
776	8	1/3	3.5	{2, 6, 3, 5}	(1.5708, 0)	3.14159	00001111 00000110 00000100 00000010 10000001 11100001 11010001 10001110	{40, 344}
777	8	1/3	3	{0, 5, 6, 3}	(0, 0)	1.0472	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 52}
778	8	1/3	3	{5, 0, 3, 6}	(0, 0)	-1.0472	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 52}
779	8	1/3	3	{5, 0, 6, 3}	(1.5708, 0.523599)	-3.14159	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 52}
780	8	1/3	3	{0, 5, 3, 6}	(1.5708, -0.523599)	3.14159	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 52}
781	8	1/3	22	{2, 3, 2, 3}	(0, 0)	0	00001110 00000101 00000011 00000011 10000101 11001001 10110000 01111100	{4, 124}
782	8	1/3	22	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001110 00000101 00000011 00000011 10000101 11001001 10110000 01111100	{4, 124}
783	8	1/3	2.75	{1, 3, 4, 2}	(1.5708, -2.0944)	-1.0472	00011100 00010111 00001111 11001011 10110011 11100001 01111000 01111100	{2, 8}
784	8	1/3	2.75	{3, 1, 2, 4}	(1.5708, 2.0944)	-1.0472	00011100 00010111 00001111 11001011 10110011 11100001 01111000 01111100	{2, 8}
785	8	1/3	2.75	{3, 1, 4, 2}	(1.10715, 3.14159)	2.63623	00011100 00010111 00001111 11001011 10110011 11100001 01111000 01111100	{2, 8}
786	8	1/3	2.75	{1, 3, 2, 4}	(1.10715, 0)	-2.63623	00011100 00010111 00001111 11001011 10110011 11100001 01111000 01111100	{2, 8}
787	8	1/3	2.5	{7, 2, 6, 2}	(0, 0)	-3.14159	00010110 00001011 00000100 10000001 01000011 10100001 11001000 01011100	{8, 128}
788	8	1/3	2.5	{4, 0, 6, 5}	(0.588003, -2.0944)	-2.24593	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
789	8	1/3	2.5	{0, 4, 5, 6}	(0.588003, -1.0472)	2.24593	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
790	8	1/3	2.5	{5, 4, 0, 6}	(0.857072, -2.61799)	-1.44547	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
791	8	1/3	2.5	{6, 5, 4, 0}	(0.588003, 2.0944)	-2.24593	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
792	8	1/3	2.5	{5, 6, 0, 4}	(0.588003, 1.0472)	2.24593	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
793	8	1/3	2.5	{0, 6, 5, 4}	(0.857072, 2.61799)	-1.44547	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
794	8	1/3	2.5	{6, 0, 4, 5}	(0.857072, 0.523599)	1.44547	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
795	8	1/3	2.5	{4, 5, 6, 0}	(0.857072, -0.523599)	1.44547	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
796	8	1/3	2.5	{4, 5, 0, 6}	(1.28976, 2.61799)	2.24593	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
797	8	1/3	2.5	{0, 6, 4, 5}	(1.28976, -2.61799)	2.24593	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
798	8	1/3	2.5	{6, 0, 5, 4}	(1.28976, -0.523599)	-2.24593	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
799	8	1/3	2.5	{6, 5, 0, 4}	(1.10715, -2.0944)	2.63623	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
800	8	1/3	2.5	{5, 6, 4, 0}	(1.10715, -1.0472)	-2.63623	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
801	8	1/3	2.5	{5, 4, 6, 0}	(1.28976, 0.523599)	-2.24593	00011000 00001111 00000111 10000011 11000001 01100000 01110000 01111000	{1, 3}
802	8	1/3	2.5	{0, 4, 6, 5}	(1.10715, 2.0944)	2.63623	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
803	8	1/3	2.5	{4, 0, 5, 6}	(1.10715, 1.0472)	-2.63623	00001110 00001011 00000111 00000011 11000101 10101000 11110000 01111000	{1, 3}
804	8	1/3	2.5	{2, 7, 6, 2}	(1.5708, -1.5708)	3.14159	00010110 00001011 00000100 10000001 01000011 10100001 11001000 01011100	{8, 128}
805	8	1/3	2.4	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00000111 00000010 11000000 11100001 11110000 11100100	{4, 48}
806	8	1/3	2.4	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001111 00001111 00000111 00000010 11000000 11100001 11110000 11100100	{4, 48}
807	8	1/3	2.33333	{0, 3, 2, 1}	(0, 0)	0	00001110 00001101 00001011 00000111 11100010 11010001 10111000 01110100	{16, 48}
808	8	1/3	2.33333	{0, 3, 1, 2}	(1.5708, 0)	-3.14159	00001110 00001101 00001011 00000111 11100010 11010001 10111000 01110100	{16, 48}
809	8	1/3	2.25	{6, 7, 6, 7}	(0, 0)	0	00010111 00001111 00000011 10000111 01000111 11011000 11111000 11111000	{4, 84}
810	8	1/3	2.25	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	00010111 00001111 00000011 10000111 01000111 11011000 11111000 11111000	{4, 84}
811	8	1/3	2	{1, 2, 4, 3}	(0.588003, -3.14159)	2.24593	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{2, 8}
812	8	1/3	2	{0, 3, 1, 2}	(1.5708, 2.0944)	1.0472	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
813	8	1/3	2	{3, 0, 2, 1}	(1.5708, -2.0944)	1.0472	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 4}
814	8	1/3	2	{2, 1, 3, 4}	(0.588003, 0)	-2.24593	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{2, 8}
815	8	1/3	2	{3, 0, 1, 2}	(1.10715, -3.14159)	-2.63623	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 4}
816	8	1/3	2	{2, 1, 4, 3}	(1.5708, -2.0944)	-2.0944	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{2, 8}
817	8	1/3	2	{1, 2, 3, 4}	(1.5708, 2.0944)	-2.0944	00010110 00001010 00000110 10000001 01000001 10100001 11100001 00011110	{2, 8}
818	8	1/3	2	{0, 3, 2, 1}	(1.10715, 0)	2.63623	00001011 00001010 00000111 00000101 11000000 00110000 11100000 10110000	{4, 4}
819	8	1/3	16	{0, 1, 0, 1}	(0, 0)	0	00001011 00001011 00000110 00000101 11000101 00111001 11100000 11011100	{12, 454}
820	8	1/3	16	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001011 00001011 00000110 00000101 11000101 00111001 11100000 11011100	{12, 454}
821	8	1/3	14	{1, 2, 1, 2}	(0, 0)	0	00001011 00000101 00000101 00000001 10000010 01100010 10001100 11110000	{12, 292}
822	8	1/3	14	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001011 00000101 00000101 00000001 10000010 01100010 10001100 11110000	{12, 292}
823	8	1/3	13	{0, 5, 4, 1}	(0, 0)	0	00001011 00000111 00000001 00000001 10000011 01000011 11001100 11111100	{8, 76}
824	8	1/3	13	{0, 5, 1, 4}	(1.5708, 0)	3.14159	00001011 00000111 00000001 00000001 10000011 01000011 11001100 11111100	{8, 76}
825	8	1/3	12	{6, 7, 6, 7}	(0, 0)	0	00000111 00000100 00000011 00000011 00000011 11000011 10111100 10111100	{28, 844}
826	8	1/3	12	{6, 7, 7, 6}	(1.5708, 0)	3.14159	00000111 00000100 00000011 00000011 00000011 11000011 10111100 10111100	{28, 844}
827	8	1/3	10	{2, 3, 2, 3}	(0, 0)	0	00001011 00000101 00000010 00000010 10000001 01000000 10110001 11001010	{28, 936}
828	8	1/3	10	{0, 5, 2, 5}	(0, 0)	-3.14159	00010101 00001011 00000111 10000010 01000001 10100000 01110000 11101000	{11, 122}
829	8	1/3	10	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000101 00000010 00000010 10000001 01000000 10110001 11001010	{28, 936}
830	8	1/3	10	{5, 0, 2, 5}	(1.5708, -1.5708)	3.14159	00010101 00001011 00000111 10000010 01000001 10100000 01110000 11101000	{11, 122}
831	8	1/3	1.85714	{6, 0, 2, 3}	(1.5708, 1.0472)	1.42745	00011000 00001111 00000110 10000011 11000001 01100001 01110000 01011100	{2, 2}
832	8	1/3	1.85714	{0, 6, 3, 2}	(1.5708, -1.0472)	1.42745	00011000 00001111 00000110 10000011 11000001 01100001 01110000 01011100	{2, 2}
833	8	1/3	1.85714	{0, 6, 2, 3}	(0.927295, 3.14159)	2.47465	00011000 00001111 00000110 10000011 11000001 01100001 01110000 01011100	{2, 2}
834	8	1/3	1.85714	{6, 0, 3, 2}	(0.927295, 0)	-2.47465	00011000 00001111 00000110 10000011 11000001 01100001 01110000 01011100	{2, 2}
835	8	1/3	1.77778	{2, 3, 2, 3}	(0, 0)	0	00001101 00001011 00000111 00000111 11000010 10110001 01111000 11110100	{40, 452}
836	8	1/3	1.77778	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001101 00001011 00000111 00000111 11000010 10110001 01111000 11110100	{40, 452}
837	8	1/3	1.625	{2, 3, 2, 3}	(0, 0)	0	00001101 00001010 00000111 00000111 11000010 10110001 01111000 10110100	{8, 148}
838	8	1/3	1.625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001101 00001010 00000111 00000111 11000010 10110001 01111000 10110100	{8, 148}
839	8	1/3	1.33333	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000001 00000001 10000000 11100000 11111000	{12, 224}
840	8	1/3	1.33333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000001 00000001 10000000 11100000 11111000	{12, 224}
841	8	1/3	1.125	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000011 00000011 11000011 11111101 11111110	{8, 256}
842	8	1/3	1.125	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000011 11000011 11111101 11111110	{8, 256}
843	8	1/3	1.11111	{2, 3, 2, 3}	(0, 0)	0	00001011 00000111 00000001 00000001 10000010 01000001 11001000 11110100	{24, 316}
844	8	1/3	1.11111	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000111 00000001 00000001 10000010 01000001 11001000 11110100	{24, 316}
845	8	1/3	1	{3, 2, 3, 6}	(0, 0)	1.0472	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{3, 54}
846	8	1/3	1	{2, 3, 6, 3}	(0, 0)	-1.0472	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{3, 54}
847	8	1/3	1	{2, 3, 3, 6}	(1.5708, 0.523599)	-3.14159	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{3, 54}
848	8	1/3	1	{3, 2, 6, 3}	(1.5708, -0.523599)	3.14159	00001001 00000101 00000010 00000001 10000000 01000000 00100000 11010000	{3, 54}
849	8	1/3	0.625	{2, 3, 2, 3}	(0, 0)	0	00001010 00000111 00000001 00000001 10000010 01000001 11001000 01110100	{24, 372}
850	8	1/3	0.625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001010 00000111 00000001 00000001 10000010 01000001 11001000 01110100	{24, 372}
851	8	1/3	0.444444	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000001 00000001 10000011 11100100 11111100	{12, 120}
852	8	1/3	0.444444	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000001 00000001 10000011 11100100 11111100	{12, 120}
853	8	1/3	0.416667	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000011 10000111 11101000 11111001 11111010	{8, 48}
854	8	1/3	0.416667	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000011 10000111 11101000 11111001 11111010	{8, 48}
855	8	1/4	9.82843	{6, 7, 6, 7}	(0, 0)	0	00010111 00001111 00001111 10000011 01100000 11100000 11110000 11110000	{4, 32}
856	8	1/4	9.82843	{6, 7, 7, 6}	(1.5708, 0)	3.14159	00010111 00001111 00001111 10000011 01100000 11100000 11110000 11110000	{4, 32}
857	8	1/4	9	{0, 4, 3, 1}	(0, 0)	0	00010011 00001011 00000101 10000011 01000011 00100001 11011001 11111110	{8, 32}
858	8	1/4	9	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	00010011 00001011 00000101 10000011 01000011 00100001 11011001 11111110	{8, 32}
859	8	1/4	8	{0, 1, 0, 1}	(0, 0)	0	00000011 00000011 00000010 00000001 00000001 00000001 11100000 11011100	{24, 332}
860	8	1/4	8	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000011 00000011 00000010 00000001 00000001 00000001 11100000 11011100	{24, 332}
861	8	1/4	7.82843	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000010 00000010 10000001 01000001 00110001 11001110	{8, 32}
862	8	1/4	7.82843	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000010 00000010 10000001 01000001 00110001 11001110	{8, 32}
863	8	1/4	6	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000001 00000001 10000010 11100100 11111000	{20, 196}
864	8	1/4	6	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000001 00000001 10000010 11100100 11111000	{20, 196}
865	8	1/4	5.37258	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000101 00000010 00000001 11100001 11010000 11101100	{4, 40}
866	8	1/4	5.37258	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000101 00000010 00000001 11100001 11010000 11101100	{4, 40}
867	8	1/4	5	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000000 01000000 00110000 11100000	{44, 136}
868	8	1/4	5	{0, 5, 4, 3}	(0, 0)	-3.14159	00001101 00000110 00000011 00000001 10000000 11000000 01100000 10110000	{8, 20}
869	8	1/4	5	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000000 01000000 00110000 11100000	{44, 136}
870	8	1/4	5	{5, 0, 4, 3}	(1.5708, -1.5708)	3.14159	00001101 00000110 00000011 00000001 10000000 11000000 01100000 10110000	{8, 20}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
871	8	1/4	40.9706	{5, 6, 5, 6}	(0, 0)	0	00101111 00010111 10001111 01000111 10100111 11111001 11111001 11111110	{4, 20}
872	8	1/4	40.9706	{5, 6, 6, 5}	(1.5708, 0)	3.14159	00101111 00010111 10001111 01000111 10100111 11111001 11111001 11111110	{4, 20}
873	8	1/4	4.68629	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000001 10000010 11100001 11101000 11110100	{12, 152}
874	8	1/4	4.68629	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000001 10000010 11100001 11101000 11110100	{12, 152}
875	8	1/4	4.41421	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000011 10000001 01000001 00110000 11111100	{8, 32}
876	8	1/4	4.41421	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000011 10000001 01000001 00110000 11111100	{8, 32}
877	8	1/4	4.34315	{1, 2, 1, 2}	(0, 0)	0	00011111 00001111 00001111 10000010 11100001 11100001 11110000 11101100	{4, 32}
878	8	1/4	4.34315	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011111 00001111 00001111 10000010 11100001 11100001 11110000 11101100	{4, 32}
879	8	1/4	4	{7, 1, 7, 2}	(0, 0)	1.5708	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
880	8	1/4	4	{1, 7, 2, 7}	(0, 0)	-1.5708	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
881	8	1/4	4	{1, 7, 7, 2}	(1.5708, 0.785398)	-3.14159	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
882	8	1/4	4	{7, 1, 2, 7}	(1.5708, -0.785398)	3.14159	00000110 00000101 00000101 00000011 00000011 11100000 10011000 01111000	{8, 38}
883	8	1/4	31.3137	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000101 00000011 00000010 11100001 11011000 11110100	{4, 40}
884	8	1/4	31.3137	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000101 00000011 00000010 11100001 11011000 11110100	{4, 40}
885	8	1/4	3.37258	{5, 6, 5, 6}	(0, 0)	0	00001111 00000110 00000110 00000001 10000001 11100001 11100001 10011110	{4, 28}
886	8	1/4	3.37258	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	00001111 00000110 00000110 00000001 10000001 11100001 11100001 10011110	{4, 28}
887	8	1/4	3.34315	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000001 01000001 00110001 11101110	{32, 148}
888	8	1/4	3.34315	{6, 1, 3, 4}	(0, 0)	-3.14159	00001011 00000110 00000101 00000010 10000001 01100001 11010001 10101110	{8, 32}
889	8	1/4	3.34315	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000001 01000001 00110001 11101110	{32, 148}
890	8	1/4	3.34315	{1, 6, 3, 4}	(1.5708, -1.5708)	3.14159	00001011 00000110 00000101 00000010 10000001 01100001 11010001 10101110	{8, 32}
891	8	1/4	3	{1, 2, 1, 2}	(0, 0)	0	00000101 00000011 00000011 00000001 00000001 10000000 01100000 11111000	{20, 84}
892	8	1/4	3	{0, 1, 3, 4}	(1.5708, -3.14159)	1.5708	00000110 00000101 00000011 00000010 00000001 11000000 10110000 01101000	{4, 8}
893	8	1/4	3	{0, 1, 4, 3}	(1.5708, 3.14159)	-1.5708	00000110 00000101 00000011 00000010 00000001 11000000 10110000 01101000	{4, 8}
894	8	1/4	3	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000101 00000011 00000011 00000001 00000001 10000000 01100000 11111000	{20, 84}
895	8	1/4	27.3137	{1, 2, 1, 2}	(0, 0)	0	00001001 00000111 00000111 00000011 10000000 01100010 01110100 11110000	{4, 52}
896	8	1/4	27.3137	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001001 00000111 00000111 00000011 10000000 01100010 01110100 11110000	{4, 52}
897	8	1/4	2.68629	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000101 00000011 00000011 11100000 11011001 11111010	{8, 76}
898	8	1/4	2.68629	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000101 00000011 00000011 11100000 11011001 11111010	{8, 76}
899	8	1/4	2.5	{1, 2, 1, 2}	(0, 0)	0	00001011 00000111 00000111 00000011 10000000 01100000 11110000 11110000	{4, 24}
900	8	1/4	2.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001011 00000111 00000111 00000011 10000000 01100000 11110000 11110000	{4, 24}
901	8	1/4	2.17157	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000011 10000001 01000001 00110001 11111110	{12, 72}
902	8	1/4	2.17157	{0, 1, 2, 3}	(1.5708, -3.14159)	1.5708	00001101 00001011 00000101 00000011 11000000 10100001 01010001 11110110	{4, 8}
903	8	1/4	2.17157	{0, 1, 3, 2}	(1.5708, 3.14159)	-1.5708	00001101 00001011 00000101 00000011 11000000 10100001 01010001 11110110	{4, 8}
904	8	1/4	2.17157	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000011 10000001 01000001 00110001 11111110	{12, 72}
905	8	1/4	2.05887	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000010 00000001 10000001 11110000 11101100	{4, 56}
906	8	1/4	2.05887	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000010 00000001 10000001 11110000 11101100	{4, 56}
907	8	1/4	2	{5, 1, 5, 2}	(0, 0)	1.5708	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
908	8	1/4	2	{1, 5, 2, 5}	(0, 0)	-1.5708	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
909	8	1/4	2	{1, 5, 5, 2}	(1.5708, 0.785398)	-3.14159	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
910	8	1/4	2	{5, 1, 2, 5}	(1.5708, -0.785398)	3.14159	00000101 00000010 00000010 00000001 00000001 10000000 01100000 10011000	{4, 24}
911	8	1/4	19.6569	{0, 1, 0, 1}	(0, 0)	0	00001111 00001111 00001101 00000011 11100000 11100000 11010001 11110010	{4, 28}
912	8	1/4	19.6569	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001111 00001111 00001101 00000011 11100000 11100000 11010001 11110010	{4, 28}
913	8	1/4	18.3431	{6, 7, 6, 7}	(0, 0)	0	00010111 00001011 00000111 10000111 01000011 10110011 11111100 11111100	{4, 20}
914	8	1/4	18.3431	{6, 7, 7, 6}	(1.5708, 0)	3.14159	00010111 00001011 00000111 10000111 01000011 10110011 11111100 11111100	{4, 20}
915	8	1/4	15.6569	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000001 00000001 11000010 11100100 11111000	{4, 36}
916	8	1/4	15.6569	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000001 00000001 11000010 11100100 11111000	{4, 36}
917	8	1/4	14.6569	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000001 01000001 00110000 11101100	{16, 72}
918	8	1/4	14.6569	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000001 01000001 00110000 11101100	{16, 72}
919	8	1/4	12	{1, 2, 1, 2}	(0, 0)	0	00000101 00000011 00000011 00000010 00000001 10000000 01110000 11101000	{8, 92}
920	8	1/4	12	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000101 00000011 00000011 00000010 00000001 10000000 01110000 11101000	{8, 92}
921	8	1/4	10	{1, 2, 1, 2}	(0, 0)	0	00001001 00000111 00000111 00000001 10000000 01100000 01100000 11110000	{12, 56}
922	8	1/4	10	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001001 00000111 00000111 00000001 10000000 01100000 01100000 11110000	{12, 56}
923	8	1/4	1.91912	{0, 2, 3, 1}	(1.5708, 0)	1.23096	00001010 00001001 00000110 00000101 11000011 00110011 10101101 01011110	{4, 8}
924	8	1/4	1.91912	{0, 2, 1, 3}	(1.5708, 0)	-1.91063	00001010 00001001 00000110 00000101 11000011 00110011 10101101 01011110	{4, 8}
925	8	1/4	1.8	{0, 1, 7, 6}	(1.5708, -3.14159)	-0.927295	00001110 00001101 00001011 00000111 11100000 11010000 10110000 01110000	{24, 24}
926	8	1/4	1.8	{0, 1, 6, 7}	(1.5708, -3.14159)	2.2143	00001110 00001101 00001011 00000111 11100000 11010000 10110000 01110000	{24, 24}
927	8	1/4	1.71573	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000001 10000001 11100000 11100000 11111000	{8, 52}
928	8	1/4	1.71573	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000001 10000001 11100000 11100000 11111000	{8, 52}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
929	8	1/4	1.68629	{0, 5, 4, 1}	(0, 0)	0	00001001 00000101 00000011 00000010 10000000 01000000 00110001 11100010	{8, 48}
930	8	1/4	1.68629	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	00001001 00000101 00000011 00000010 10000000 01000000 00110001 11100010	{8, 48}
931	8	1/4	1.5	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000011 00000011 11000000 11111000 11111000	{4, 24}
932	8	1/4	1.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000011 11000000 11111000 11111000	{4, 24}
933	8	1/4	1.37258	{2, 3, 2, 3}	(0, 0)	0	00001011 00000101 00000011 00000011 10000001 01000000 10110000 11111000	{16, 164}
934	8	1/4	1.37258	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001011 00000101 00000011 00000011 10000001 01000000 10110000 11111000	{16, 164}
935	8	1/4	1.34315	{0, 1, 0, 1}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000000 11110001 11111010	{12, 76}
936	8	1/4	1.34315	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000000 11110001 11111010	{12, 76}
937	8	1/4	1.17157	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000101 10000011 11110000 11101001 11111010	{12, 108}
938	8	1/4	1.17157	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000101 10000011 11110000 11101001 11111010	{12, 108}
939	8	1/4	1.02944	{1, 2, 1, 2}	(0, 0)	0	00000111 00000011 00000011 00000001 00000001 10000001 11100000 11111100	{4, 40}
940	8	1/4	1.02944	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00000111 00000011 00000011 00000001 00000001 10000001 11100000 11111100	{4, 40}
941	8	1/4	0.857864	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000011 10000011 11100000 11111000 11111000	{4, 28}
942	8	1/4	0.857864	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000011 10000011 11100000 11111000 11111000	{4, 28}
943	8	1/4	0.804041	{1, 2, 1, 2}	(0, 0)	0	00001111 00000111 00000111 00000001 10000011 11100000 11101001 11111010	{4, 40}
944	8	1/4	0.804041	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00001111 00000111 00000111 00000001 10000011 11100000 11101001 11111010	{4, 40}
945	8	1/4	0.745166	{1, 2, 1, 2}	(0, 0)	0	00011111 00001111 00001111 10000111 11100000 11110001 11110001 11110110	{4, 28}
946	8	1/4	0.745166	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	00011111 00001111 00001111 10000111 11100000 11110001 11110001 11110110	{4, 28}
947	8	1/5	5.52786	{2, 3, 2, 3}	(0, 0)	0	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{4, 36}
948	8	1/5	5.52786	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000100 00000100 00000011 00000011 00000001 11000001 00110000 00111100	{4, 36}
949	8	1/5	4.22291	{0, 1, 0, 1}	(0, 0)	0	00001011 00001011 00000110 00000100 11000000 00110001 11100001 11000110	{4, 56}
950	8	1/5	4.22291	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	00001011 00001011 00000110 00000100 11000000 00110001 11100001 11000110	{4, 56}
951	8	1/5	3.76393	{2, 6, 5, 3}	(0, 0)	0	00001001 00000110 00000100 00000010 10000001 01100001 01010001 10001110	{8, 88}
952	8	1/5	3.76393	{2, 6, 3, 5}	(1.5708, 0)	3.14159	00001001 00000110 00000100 00000010 10000001 01100001 01010001 10001110	{8, 88}
953	8	1/5	3	{2, 5, 4, 3}	(0, 0)	0	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 16}
954	8	1/5	3	{2, 5, 3, 4}	(1.5708, 0)	-3.14159	00001010 00000101 00000010 00000001 10000000 01000000 10100000 01010000	{8, 16}
955	8	1/5	2.21115	{2, 3, 2, 3}	(0, 0)	0	00001110 00001000 00000111 00000111 11000011 10110001 10111000 00111100	{4, 36}
956	8	1/5	2.21115	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001110 00001000 00000111 00000111 11000011 10110001 10111000 00111100	{4, 36}
957	8	1/5	2.11146	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{4, 56}
958	8	1/5	2.11146	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000001 11000010 10110100 10111000	{4, 56}
959	8	1/5	11	{0, 4, 3, 1}	(0, 0)	0	00010011 00001011 00000101 10000011 01000011 00100001 11011000 11111100	{8, 48}
960	8	1/5	11	{0, 4, 1, 3}	(1.5708, 0)	3.14159	00010011 00001011 00000101 10000011 01000011 00100001 11011000 11111100	{8, 48}
961	8	1/5	1.38197	{2, 3, 2, 3}	(0, 0)	0	00000111 00000100 00000011 00000011 00000001 11000001 10110000 10111100	{8, 104}
962	8	1/5	1.38197	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000100 00000011 00000011 00000001 11000001 10110000 10111100	{8, 104}
963	8	1/5	1.22841	{2, 3, 2, 3}	(0, 0)	0	00001111 00001010 00000111 00000111 11000001 10110000 11110001 10111010	{4, 44}
964	8	1/5	1.22841	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001111 00001010 00000111 00000111 11000001 10110000 11110001 10111010	{4, 44}
965	8	1/5	0.844582	{2, 3, 2, 3}	(0, 0)	0	00001111 00001011 00000111 00000111 11000110 10111001 11111000 11110100	{4, 28}
966	8	1/5	0.844582	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00001111 00001011 00000111 00000111 11000110 10111001 11111000 11110100	{4, 28}
967	8	1/5	0.806504	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 36}
968	8	1/5	0.806504	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000010 11110100 11111000	{4, 36}
969	8	1/5	0.614207	{2, 3, 2, 3}	(0, 0)	0	00000111 00000111 00000011 00000011 00000001 11000001 11110000 11111100	{4, 32}
970	8	1/5	0.614207	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	00000111 00000111 00000011 00000011 00000001 11000001 11110000 11111100	{4, 32}
971	9	4/5	94.7214	{1, 2, 1, 2}	(0, 0)	0	000011011 000000111 000000111 000000100 100001011 100010001 011100000 111010001 111011010	{4}
972	9	4/5	94.7214	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000000111 000000111 000000100 100001011 100010001 011100000 111010001 111011010	{4}
973	9	4/5	90.2492	{1, 2, 1, 2}	(0, 0)	0	000010011 000001101 000001101 000000110 100000010 011000111 011101001 100111001 111001110	{8}
974	9	4/5	90.2492	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001101 000001101 000000110 100000010 011000111 011101001 100111001 111001110	{8}
975	9	4/5	9.77139	{1, 2, 5, 4}	(1.5708, 0)	-0.96804	000100101 000010110 000001110 100000101 010000011 001000011 111100011 011011101 100111110	{4}
976	9	4/5	9.77139	{1, 2, 4, 5}	(1.5708, 0)	2.17355	000100101 000010110 000001110 100000101 010000011 001000011 111100011 011011101 100111110	{4}
977	9	4/5	9.73607	{6, 7, 6, 7}	(0, 0)	0	000011111 000001111 000000111 000000110 100000111 110000111 111111001 111111001 111011110	{16}
978	9	4/5	9.73607	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000001111 000000111 000000110 100000111 110000111 111111001 111111001 111011110	{16}
979	9	4/5	9.34752	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000110 000000101 110000011 001110001 111101001 111011110	{8}
980	9	4/5	9.34752	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000110 000000101 110000011 001110001 111101001 111011110	{8}
981	9	4/5	83.0132	{1, 2, 1, 2}	(0, 0)	0	000101100 000011111 000011111 100000011 011000110 111000001 111010000 011110001 011101010	{8}
982	9	4/5	83.0132	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101100 000011111 000011111 100000011 011000110 111000001 111010000 011110001 011101010	{8}
983	9	4/5	8.1305	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001100 100000011 011100111 111101001 111011000 111011100	{4}
984	9	4/5	8.1305	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000001100 100000011 011100111 111101001 111011000 111011100	{4}
985	9	4/5	75.7771	{7, 8, 7, 8}	(0, 0)	0	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{32}
986	9	4/5	75.7771	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{32}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
987	9	4/5	7.63932	{1, 2, 1, 2}	(0, 0)	0	000010101 000001111 000001111 000000011 100000011 011000001 111000010 011110100 111111000	{20}
988	9	4/5	7.63932	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001111 000001111 000000011 100000011 011000001 111000010 011110100 111111000	{20}
989	9	4/5	66.732	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000001101 100000111 111100001 011110001 111010001 111111110	{4}
990	9	4/5	66.732	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000001101 100000111 111100001 011110001 111010001 111111110	{4}
991	9	4/5	6.89443	{2, 3, 2, 3}	(0, 0)	0	000010111 000001101 000001011 000001011 100000100 011100011 110010010 101101101 111101010	{4}
992	9	4/5	6.89443	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001101 000001011 000001011 100000100 011100011 110010010 101101101 111101010	{4}
993	9	4/5	6.68328	{2, 3, 2, 3}	(0, 0)	0	000001111 000001101 000000011 000000011 000000010 110000101 110001001 101110000 111101100	{16}
994	9	4/5	6.68328	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000001101 000000011 000000011 000000010 110000101 110001001 101110000 111101100	{16}
995	9	4/5	6.58359	{2, 3, 2, 3}	(0, 0)	0	000011011 000001110 000000011 000000011 100000101 110000000 010010001 111100000 101110100	{28}
996	9	4/5	6.58359	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000001110 000000011 000000011 100000101 110000000 010010001 111100000 101110100	{28}
997	9	4/5	6.52786	{0, 4, 3, 1}	(0, 0)	0	000101111 000011111 000000011 100001111 010001111 110110010 110110001 111111000 111110100	{16}
998	9	4/5	6.52786	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	000101111 000011111 000000011 100001111 010001111 110110010 110110001 111111000 111110100	{16}
999	9	4/5	6.34164	{1, 2, 1, 2}	(0, 0)	0	000010101 000001111 000001111 000001011 100000011 011100110 111001011 011111101 111110110	{4}
1000	9	4/5	6.34164	{1, 2, 2, 1}	(1.5708, 0)	3.14159	000010101 000001111 000001111 000001011 100000011 011100110 111001011 011111101 111110110	{4}
1001	9	4/5	52.3607	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001000 100000011 011100000 111000001 111010001 111010110	{4}
1002	9	4/5	52.3607	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000001000 100000011 011100000 111000001 111010001 111010110	{4}
1003	9	4/5	5.93112	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000110 000000001 110000011 001100001 111101001 111011110	{8}
1004	9	4/5	5.93112	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000110 000000001 110000011 001100001 111101001 111011110	{8}
1005	9	4/5	5.78885	{6, 7, 6, 7}	(0, 0)	0	000011111 000010110 000001111 000001111 110000000 101100001 111100001 111100001 101101110	{4}
1006	9	4/5	5.78885	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000010110 000001111 000001111 110000000 101100001 111100001 111100001 101101110	{4}
1007	9	4/5	5.26393	{6, 7, 6, 7}	(0, 0)	0	000011111 000001111 000001110 000000111 100001111 111010111 111111001 111111001 110111110	{16}
1008	9	4/5	5.26393	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000001111 000001110 000000111 100001111 111010111 111111001 111111001 110111110	{16}
1009	9	4/5	5.1459	{0, 4, 5, 1}	(0, 0)	0	000010010 000001010 000000110 000000011 100000001 010000001 001000001 111100001 000111110	{32}
1010	9	4/5	5.1459	{0, 4, 1, 5}	(1.5708, 0)	-3.14159	000010010 000001010 000000110 000000011 100000001 010000001 001000001 111100001 000111110	{32}
1011	9	4/5	47.8885	{2, 3, 2, 3}	(0, 0)	0	000011101 000001100 000000111 000000111 100000011 110000010 111100001 001111001 101110110	{12}
1012	9	4/5	47.8885	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000001100 000000111 000000111 100000011 110000010 111100001 001111001 101110110	{12}
1013	9	4/5	45.1246	{2, 3, 2, 3}	(0, 0)	0	000011001 000001010 000000101 000000101 100000011 110000000 001100010 010010100 101110000	{24}
1014	9	4/5	45.1246	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011001 000001010 000000101 000000101 100000011 110000000 001100010 010010100 101110000	{24}
1015	9	4/5	43.4164	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001011 110001001 001110010 111000010 110101100 111110000	{8}
1016	9	4/5	43.4164	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001011 110001001 001110010 111000010 110101100 111110000	{8}
1017	9	4/5	40.6525	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{4}
1018	9	4/5	40.6525	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{4}
1019	9	4/5	4.87539	{2, 3, 2, 3}	(0, 0)	0	000011001 000001011 000000101 000000101 100000010 110000000 001100010 010010100 111100000	{24}
1020	9	4/5	4.87539	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011001 000001011 000000101 000000101 100000010 110000000 001100010 010010100 111100000	{24}
1021	9	4/5	4.83688	{6, 7, 6, 7}	(0, 0)	0	000111111 000010111 000001111 100011111 110101111 101110111 111111001 111111001 111111110	{4}
1022	9	4/5	4.83688	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000111111 000010111 000001111 100011111 110101111 101110111 111111001 111111001 111111110	{4}
1023	9	4/5	4.82405	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000110 000000101 000000011 110000111 111101011 111011101 110111110	{8}
1024	9	4/5	4.82405	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000110 000000101 000000011 110000111 111101011 111011101 110111110	{8}
1025	9	4/5	4.58018	{0, 2, 5, 6}	(1.5708, 3.14159)	-1.40974	000111101 000101111 000011111 110010101 101100011 111000011 111100001 011011001 111111110	{4}
1026	9	4/5	4.58018	{0, 2, 6, 5}	(1.5708, -3.14159)	1.73185	000111101 000101111 000011111 110010101 101100011 111000011 111100001 011011001 111111110	{4}
1027	9	4/5	4.22291	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000011 100000000 011000000 011000000 111100000 111100000	{12}
1028	9	4/5	4.22291	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000011 100000000 011000000 011000000 111100000 111100000	{12}
1029	9	4/5	4	{1, 2, 1, 2}	(0, 0)	0	000010010 000001111 000001111 000001100 100000010 011100101 011101001 111010001 011001110	{4}
1030	9	4/5	4	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010010 000001111 000001111 000001100 100000010 011100101 011101001 111010001 011001110	{4}
1031	9	4/5	36.1803	{1, 2, 1, 2}	(0, 0)	0	000011011 000000111 000000111 000000001 100001001 100010001 011000010 111000100 111111000	{8}
1032	9	4/5	36.1803	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000000111 000000111 000000001 100001001 100010001 011000010 111000100 111111000	{8}
1033	9	4/5	358.885	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001001 100000111 011100001 111010011 111010101 111111110	{4}
1034	9	4/5	358.885	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000001001 100000111 011100001 111010011 111010101 111111110	{4}
1035	9	4/5	34.4721	{2, 3, 2, 3}	(0, 0)	0	000011111 000001100 000000111 000000111 100001001 110010011 111100011 101101101 101111110	{4}
1036	9	4/5	34.4721	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001100 000000111 000000111 100001001 110010011 111100011 101101101 101111110	{4}
1037	9	4/5	33.4164	{2, 3, 2, 3}	(0, 0)	0	000001101 000001100 000000011 000000011 000000010 110000101 110001001 001110000 101101100	{48}
1038	9	4/5	33.4164	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001100 000000011 000000011 000000010 110000101 110001001 001110000 101101100	{48}
1039	9	4/5	31.7082	{1, 2, 1, 2}	(0, 0)	0	000010110 000001111 000001111 000000001 100000011 011000101 111001011 111010101 011111110	{4}
1040	9	4/5	31.7082	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010110 000001111 000001111 000000001 100000011 011000101 111001011 111010101 011111110	{4}
1041	9	4/5	309.289	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000001010 100000111 111100011 111010000 011111001 111011010	{4}
1042	9	4/5	309.289	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000001010 100000111 111100011 111010000 011111001 111011010	{4}
1043	9	4/5	3.71885	{1, 2, 1, 2}	(0, 0)	0	000010100 000001111 000001111 000001000 100000010 011100001 111000001 011010001 011001110	{8}
1044	9	4/5	3.71885	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010100 000001111 000001111 000001000 100000010 011100001 111000001 011010001 011001110	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1045	9	4/5	3.525	{2, 3, 2, 3}	(0, 0)	0	000011111 000011111 000001111 000001111 110000011 111100100 111101000 111110001 111110010	{4}
1046	9	4/5	3.525	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001111 110000011 111100100 111101000 111110001 111110010	{4}
1047	9	4/5	3.44721	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000011 100000111 111000101 011011001 111110000 111111100	{16}
1048	9	4/5	3.44721	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000011 100000111 111000101 011011001 111110000 111111100	{16}
1049	9	4/5	3.12868	{6, 7, 6, 7}	(0, 0)	0	000011111 000001111 000000111 000000111 100000111 110000111 111111001 111111001 111111110	{4}
1050	9	4/5	3.12868	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000001111 000000111 000000111 100000111 110000111 111111001 111111001 111111110	{4}
1051	9	4/5	28.9443	{3, 4, 3, 4}	(0, 0)	0	000001011 000001010 000000110 000000101 000000101 110000010 001110001 111001001 100110110	{28}
1052	9	4/5	28.9443	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001011 000001010 000000110 000000101 000000101 110000010 001110001 111001001 100110110	{28}
1053	9	4/5	27.2361	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000110 000000101 110000111 001111001 111101001 111011110	{8}
1054	9	4/5	27.2361	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000110 000000101 110000111 001111001 111101001 111011110	{8}
1055	9	4/5	24.4721	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001111 000001011 110000000 001100011 111000001 111101001 111101110	{8}
1056	9	4/5	24.4721	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001111 000001011 110000000 001100011 111000001 111101001 111101110	{8}
1057	9	4/5	24.3713	{6, 7, 6, 7}	(0, 0)	0	000111111 000011111 000000110 100001111 110001111 110110111 111111001 111111001 110111110	{4}
1058	9	4/5	24.3713	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000111111 000011111 000000110 100001111 110001111 110110111 111111001 111111001 110111110	{4}
1059	9	4/5	20.9443	{1, 4, 2, 3}	(0, 0)	0	000001111 000000101 000000101 000000011 000000011 100000000 111000001 100110001 111110110	{16}
1060	9	4/5	20.9443	{1, 4, 3, 2}	(1.5708, 0)	3.14159	000001111 000000101 000000101 000000011 000000011 100000000 111000001 100110001 111110110	{16}
1061	9	4/5	20	{5, 7, 5, 7}	(0, 0)	0	000001110 000001011 000001011 000000101 000000101 111000001 100110001 111000001 011111110	{36}
1062	9	4/5	20	{5, 7, 7, 5}	(1.5708, 0)	3.14159	000001110 000001011 000001011 000000101 000000101 111000001 100110001 111000001 011111110	{36}
1063	9	4/5	2.71353	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000010 000000001 000000001 100000001 110000010 101000100 110111000	{16}
1064	9	4/5	2.71353	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000010 000000001 000000001 100000001 110000010 101000100 110111000	{16}
1065	9	4/5	2.48107	{0, 1, 4, 3}	(1.5708, 3.14159)	-1.40974	000001100 000001010 000000110 000000101 000000011 110000111 101101001 011011001 000111110	{4}
1066	9	4/5	2.48107	{0, 1, 3, 4}	(1.5708, -3.14159)	1.73185	000001100 000001010 000000110 000000101 000000011 110000111 101101001 011011001 000111110	{4}
1067	9	4/5	2.21115	{3, 4, 3, 4}	(0, 0)	0	000001111 000001011 000000110 000000101 000000101 110000010 101110001 111001001 110110110	{20}
1068	9	4/5	2.21115	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001011 000000110 000000101 000000101 110000010 101110001 111001001 110110110	{20}
1069	9	4/5	2.07295	{2, 3, 2, 3}	(0, 0)	0	000001101 000000100 000000010 000000010 000000001 100000000 110000001 001100001 100010110	{32}
1070	9	4/5	2.07295	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000000100 000000010 000000010 000000001 100000000 110000001 001100001 100010110	{32}
1071	9	4/5	2.05573	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{32}
1072	9	4/5	2.05573	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{32}
1073	9	4/5	198.387	{6, 7, 6, 7}	(0, 0)	0	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{8}
1074	9	4/5	198.387	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{8}
1075	9	4/5	19.9443	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000010 100000001 010000000 001000000 100100000 111010000	{8}
1076	9	4/5	19.9443	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000010 100000001 010000000 001000000 100100000 111010000	{8}
1077	9	4/5	18.2918	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001011 110000001 001100010 111000010 110101100 111110000	{16}
1078	9	4/5	18.2918	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001011 110000001 001100010 111000010 110101100 111110000	{16}
1079	9	4/5	17.2361	{1, 2, 1, 2}	(0, 0)	0	000001011 000000101 000000101 000000011 000000011 100000010 011000001 100111001 111110110	{60}
1080	9	4/5	17.2361	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000101 000000101 000000011 000000011 100000010 011000001 100111001 111110110	{60}
1081	9	4/5	15.5279	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
1082	9	4/5	15.5279	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
1083	9	4/5	15.4721	{3, 8, 7, 4}	(0, 0)	0	000001100 000001011 000000111 000000010 000000001 110000011 101000011 011101100 011011100	{16}
1084	9	4/5	15.4721	{3, 8, 4, 7}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000010 000000001 110000011 101000011 011101100 011011100	{16}
1085	9	4/5	15.1554	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000001010 100000111 111100011 111010001 011111001 111011110	{12}
1086	9	4/5	15.1554	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000001010 100000111 111100011 111010001 011111001 111011110	{12}
1087	9	4/5	14.2082	{3, 4, 3, 4}	(0, 0)	0	000001101 000000100 000000011 000000010 000000010 100000000 110000001 001110001 101000110	{16}
1088	9	4/5	14.2082	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001101 000000100 000000011 000000010 000000010 100000000 110000001 001110001 101000110	{16}
1089	9	4/5	137.082	{1, 2, 1, 2}	(0, 0)	0	000010101 000001111 000001111 000001010 100000011 011100001 111000010 011110100 111011000	{16}
1090	9	4/5	137.082	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001111 000001111 000001010 100000011 011100001 111000010 011110100 111011000	{16}
1091	9	4/5	13.8736	{4, 5, 7, 6}	(1.5708, 0)	-0.96804	000111001 000011111 000011111 100000111 111000101 111000011 011110001 011101001 111111110	{4}
1092	9	4/5	13.8736	{4, 5, 6, 7}	(1.5708, 0)	2.17355	000111001 000011111 000011111 100000111 111000101 111000011 011110001 011101001 111111110	{4}
1093	9	4/5	13.8197	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000101 000000101 110000001 001110001 111000001 111111110	{8}
1094	9	4/5	13.8197	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000101 000000101 110000001 001110001 111000001 111111110	{8}
1095	9	4/5	13.7812	{6, 7, 6, 7}	(0, 0)	0	000011111 000010111 000001111 000000110 110001111 101010111 111111001 111111001 111011110	{4}
1096	9	4/5	13.7812	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000010111 000001111 000000110 110001111 101010111 111111001 111111001 111011110	{4}
1097	9	4/5	12.7639	{2, 3, 2, 3}	(0, 0)	0	000001011 000001010 000000101 000000101 000000001 110000011 001100000 110001001 101111010	{48}
1098	9	4/5	12.7639	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000001010 000000101 000000101 000000001 110000011 001100000 110001001 101111010	{48}
1099	9	4/5	12.6631	{7, 8, 7, 8}	(0, 0)	0	000001111 000001111 000001011 000000111 000000011 111000111 110101011 111111100 111111100	{4}
1100	9	4/5	12.6631	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001111 000001111 000001011 000000111 000000011 111000111 110101011 111111100 111111100	{4}
1101	9	4/5	12.1115	{2, 3, 2, 3}	(0, 0)	0	000011110 000001011 000000111 000000111 100000100 110000001 101110001 111100000 011101100	{4}
1102	9	4/5	12.1115	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000001011 000000111 000000111 100000100 110000001 101110001 111100000 011101100	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1103	9	4/5	118.138	{6, 7, 6, 7}	(0, 0)	0	000101111 000010111 000001110 100001111 010000001 101100001 111100001 111100001 110111110	{4}
1104	9	4/5	118.138	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000101111 000010111 000001110 100001111 010000001 101100001 111100001 111100001 110111110	{4}
1105	9	4/5	11.0557	{7, 8, 7, 8}	(0, 0)	0	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{36}
1106	9	4/5	11.0557	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{36}
1107	9	4/5	106.43	{2, 3, 2, 3}	(0, 0)	0	000001011 000001001 000000111 000000111 000000101 110000010 001110011 101101101 111110110	{8}
1108	9	4/5	106.43	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000001001 000000111 000000111 000000101 110000010 001110011 101101101 111110110	{8}
1109	9	4/5	10.4721	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000001101 100000010 011100101 011101000 111010000 111101000	{4}
1110	9	4/5	10.4721	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000001101 100000010 011100101 011101000 111010000 111101000	{4}
1111	9	4/5	1.45898	{2, 3, 2, 3}	(0, 0)	0	000011010 000001111 000000011 000000011 100000101 110000000 010010000 111100001 011110010	{16}
1112	9	4/5	1.45898	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011010 000001111 000000011 000000011 100000101 110000000 010010000 111100001 011110010	{16}
1113	9	4/5	1.40325	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100000 111101100	{8}
1114	9	4/5	1.40325	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100000 111101100	{8}
1115	9	4/5	1.03648	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000010 000000001 000000001 100000000 110000010 101000100 110110000	{16}
1116	9	4/5	1.03648	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000010 000000001 000000001 100000000 110000010 101000100 110110000	{16}
1117	9	4/5	0.791796	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000011 000000001 000000001 100000000 110000010 101000100 111110000	{16}
1118	9	4/5	0.791796	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000011 000000001 000000001 100000000 110000010 101000100 111110000	{16}
1119	9	3/4	9.24264	{2, 3, 2, 3}	(0, 0)	0	000010111 000001001 000000011 000000011 100000110 010000001 100010010 101110100 111101000	{8}
1120	9	3/4	9.24264	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001001 000000011 000000011 100000110 010000001 100010010 101110100 111101000	{8}
1121	9	3/4	8.82843	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000111 101001001 110111000 101111100	{44}
1122	9	3/4	8.82843	{0, 1, 4, 3}	(1.5708, 0)	-1.5708	000100101 000010101 000001110 100000011 010000011 001000000 111000000 001110000 110110000	{4}
1123	9	3/4	8.82843	{0, 1, 3, 4}	(1.5708, 0)	1.5708	000100101 000010101 000001110 100000011 010000011 001000000 111000000 001110000 110110000	{4}
1124	9	3/4	8.82843	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000111 101001001 110111000 101111100	{44}
1125	9	3/4	8.61204	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000100 000000011 000000011 100000001 011000011 010110101 110111110	{4}
1126	9	3/4	8.61204	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000100 000000011 000000011 100000001 011000011 010110101 110111110	{4}
1127	9	3/4	8.37534	{1, 2, 1, 2}	(0, 0)	0	000001011 000000111 000000111 000000101 000000010 100000001 011100001 111010000 111101100	{4}
1128	9	3/4	8.37534	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000111 000000111 000000101 000000010 100000001 011100001 111010000 111101100	{4}
1129	9	3/4	7.92226	{3, 4, 3, 4}	(0, 0)	0	000001001 000000110 000000101 000000011 000000011 100000000 011000011 010110100 101110100	{4}
1130	9	3/4	7.92226	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000110 000000101 000000011 000000011 100000000 011000011 010110100 101110100	{4}
1131	9	3/4	7.82843	{7, 0, 7, 5}	(0, 0)	2.35619	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
1132	9	3/4	7.82843	{0, 8, 5, 7}	(0, 0)	0.785398	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
1133	9	3/4	7.82843	{0, 7, 5, 7}	(0, 0)	-2.35619	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
1134	9	3/4	7.82843	{8, 0, 7, 5}	(0, 0)	-0.785398	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
1135	9	3/4	7.82843	{0, 7, 7, 5}	(1.5708, 1.1781)	-3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
1136	9	3/4	7.82843	{8, 0, 5, 7}	(1.5708, 0.392699)	-3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
1137	9	3/4	7.82843	{7, 0, 5, 7}	(1.5708, -1.1781)	3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
1138	9	3/4	7.82843	{0, 8, 7, 5}	(1.5708, -0.392699)	3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
1139	9	3/4	7.77124	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100001 111011110	{4}
1140	9	3/4	7.77124	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100001 111011110	{4}
1141	9	3/4	7.70069	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000000011 100000001 111000000 111000000 011100001 111110010	{4}
1142	9	3/4	7.70069	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000000011 100000001 111000000 111000000 011100001 111110010	{4}
1143	9	3/4	7.32843	{1, 2, 5, 4}	(1.5708, 3.14159)	-0.785398	000100111 000010010 000001010 100001110 010000001 001000001 100100001 111100001 100011110	{4}
1144	9	3/4	7.32843	{1, 2, 4, 5}	(1.5708, -3.14159)	2.35619	000100111 000010010 000001010 100001110 010000001 001000001 100100001 111100001 100011110	{4}
1145	9	3/4	6.34315	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000000 101000001 110110001 101110110	{20}
1146	9	3/4	6.34315	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000000 101000001 110110001 101110110	{20}
1147	9	3/4	6.32843	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000011 110000111 101101001 011111000 101111100	{12}
1148	9	3/4	6.32843	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000011 110000111 101101001 011111000 101111100	{12}
1149	9	3/4	6.28892	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000111 000000011 000000011 100000001 011000011 011110101 111111110	{4}
1150	9	3/4	6.28892	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000111 000000011 000000011 100000001 011000011 011110101 111111110	{4}
1151	9	3/4	6	{3, 8, 7, 8}	(0, 0)	0.785398	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
1152	9	3/4	6	{8, 3, 8, 7}	(0, 0)	-0.785398	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
1153	9	3/4	6	{8, 3, 7, 8}	(1.5708, 0.392699)	-3.14159	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
1154	9	3/4	6	{3, 8, 8, 7}	(1.5708, -0.392699)	3.14159	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
1155	9	3/4	5.95617	{2, 3, 2, 3}	(0, 0)	0	000000110 000000101 000000011 000000011 000000001 000000001 110000011 101100100 011111100	{4}
1156	9	3/4	5.95617	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000110 000000101 000000011 000000011 000000001 000000001 110000011 101100100 011111100	{4}
1157	9	3/4	5.88562	{4, 5, 4, 5}	(0, 0)	0	000011101 000011101 000011010 000000110 111000011 111000011 110100011 001111100 110011100	{4}
1158	9	3/4	5.88562	{4, 5, 5, 4}	(1.5708, 0)	3.14159	000011101 000011101 000011010 000000110 111000011 111000011 110100011 001111100 110011100	{4}
1159	9	3/4	5.82843	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100001 111111110	{4}
1160	9	3/4	5.82843	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100001 111111110	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1161	9	3/4	5.41421	{0, 1, 5, 4}	(1.5708, 0)	-1.5708	000010101 000001101 000000110 000000110 100000011 010000011 111100000 001111000 110011000	{4}
1162	9	3/4	5.41421	{0, 1, 4, 5}	(1.5708, 0)	1.5708	000010101 000001101 000000110 000000110 100000011 010000011 111100000 001111000 110011000	{4}
1163	9	3/4	5.37258	{3, 4, 3, 4}	(0, 0)	0	000001101 000001011 000001010 000000110 000000110 111000001 100110001 011110000 110001100	{12}
1164	9	3/4	5.37258	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001101 000001011 000001010 000000110 000000110 111000001 100110001 011110000 110001100	{12}
1165	9	3/4	5.02944	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000011010 000000111 111000011 111000011 110100001 111111001 110111110	{4}
1166	9	3/4	5.02944	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000011010 000000111 111000011 111000011 110100001 111111001 110111110	{4}
1167	9	3/4	4.77191	{0, 1, 0, 1}	(0, 0)	0	000001110 000001110 000001001 000000111 000000110 111000001 110110001 110110001 001101110	{4}
1168	9	3/4	4.77191	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001110 000001110 000001001 000000111 000000110 111000001 110110001 110110001 001101110	{4}
1169	9	3/4	4.41421	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000011 100000001 111000111 111001000 111101001 111111010	{8}
1170	9	3/4	4.41421	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000011 100000001 111000111 111001000 111101001 111111010	{8}
1171	9	3/4	4	{7, 1, 7, 6}	(0, 0)	2.35619	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
1172	9	3/4	4	{2, 8, 7, 8}	(0, 0)	0.785398	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
1173	9	3/4	4	{1, 7, 6, 7}	(0, 0)	-2.35619	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
1174	9	3/4	4	{8, 2, 8, 7}	(0, 0)	-0.785398	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
1175	9	3/4	4	{1, 7, 7, 6}	(1.5708, 1.1781)	-3.14159	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
1176	9	3/4	4	{8, 2, 7, 8}	(1.5708, 0.392699)	-3.14159	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
1177	9	3/4	4	{7, 1, 6, 7}	(1.5708, -1.1781)	3.14159	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
1178	9	3/4	4	{2, 8, 8, 7}	(1.5708, -0.392699)	3.14159	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
1179	9	3/4	36.9706	{3, 4, 3, 4}	(0, 0)	0	000001100 000001010 000000101 000000011 000000011 110000011 101000001 010111001 001111110	{24}
1180	9	3/4	36.9706	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001100 000001010 000000101 000000011 000000011 110000011 101000001 010111001 001111110	{24}
1181	9	3/4	31.3137	{2, 3, 2, 3}	(0, 0)	0	000001101 000001011 000000110 000000110 000000011 110000101 101101001 011110001 110011110	{12}
1182	9	3/4	31.3137	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001011 000000110 000000110 000000011 110000101 101101001 011110001 110011110	{12}
1183	9	3/4	3.80474	{3, 0, 5, 1}	(1.5708, -1.5708)	-1.23096	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
1184	9	3/4	3.80474	{0, 3, 1, 5}	(1.5708, 1.5708)	-1.23096	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
1185	9	3/4	3.80474	{3, 0, 1, 5}	(0.955317, 0)	-3.14159	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
1186	9	3/4	3.80474	{0, 3, 5, 1}	(0.955317, -3.14159)	-3.14159	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
1187	9	3/4	3.78361	{3, 4, 3, 4}	(0, 0)	0	000001011 000000110 000000101 000000011 000000011 100000001 011000000 110110000 101111000	{8}
1188	9	3/4	3.78361	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001011 000000110 000000101 000000011 000000011 100000001 011000000 110110000 101111000	{8}
1189	9	3/4	3.71573	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000111 000000100 110001011 110010011 111100001 111011001 111011110	{8}
1190	9	3/4	3.71573	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000111 000000100 110001011 110010011 111100001 111011001 111011110	{8}
1191	9	3/4	3.70711	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000001 110000000 101100001 011100000 101110100	{8}
1192	9	3/4	3.70711	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000001 110000000 101100001 011100000 101110100	{8}
1193	9	3/4	3.48904	{2, 3, 2, 3}	(0, 0)	0	000011101 000010010 000001111 000001111 110000000 101100001 101100001 011100000 101101100	{4}
1194	9	3/4	3.48904	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000010010 000001111 000001111 110000000 101100001 101100001 011100000 101101100	{4}
1195	9	3/4	3.41421	{4, 5, 4, 5}	(0, 0)	0	000000101 000000100 000000011 000000010 000000001 000000001 110000001 001100000 101011100	{4}
1196	9	3/4	3.41421	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000100 000000011 000000010 000000001 000000001 110000001 001100000 101011100	{4}
1197	9	3/4	3.17157	{2, 3, 2, 3}	(0, 0)	0	000010101 000001001 000000011 000000011 100000101 010000001 100010000 001100000 111111000	{20}
1198	9	3/4	3.17157	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001001 000000011 000000011 100000101 010000001 100010000 001100000 111110000	{20}
1199	9	3/4	3.09384	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000100 000000011 000000011 100000001 011000000 010110001 110111010	{4}
1200	9	3/4	3.09384	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000100 000000011 000000011 100000001 011000000 010110001 110111010	{4}
1201	9	3/4	3.02944	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000100 101001001 110110000 101110100	{16}
1202	9	3/4	3.02944	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000100 101001001 110110000 101110100	{16}
1203	9	3/4	3	{0, 7, 0, 2}	(0, 0)	2.35619	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1204	9	3/4	3	{7, 0, 2, 1}	(0, 0)	0.785398	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1205	9	3/4	3	{7, 0, 2, 0}	(0, 0)	-2.35619	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1206	9	3/4	3	{0, 7, 1, 2}	(0, 0)	-0.785398	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1207	9	3/4	3	{7, 0, 0, 2}	(1.5708, 1.1781)	-3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1208	9	3/4	3	{0, 7, 2, 1}	(1.5708, 0.392699)	-3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1209	9	3/4	3	{0, 7, 2, 0}	(1.5708, -1.1781)	3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1210	9	3/4	3	{7, 0, 1, 2}	(1.5708, -0.392699)	3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
1211	9	3/4	29.3137	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000111 000000101 110000011 110000011 111100001 111011001 111111110	{4}
1212	9	3/4	29.3137	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000111 000000101 110000011 110000011 111100001 111011001 111111110	{4}
1213	9	3/4	21.6569	{2, 3, 2, 3}	(0, 0)	0	000011011 000001001 000000111 000000111 100000010 110000001 001100011 101110100 111101100	{56}
1214	9	3/4	21.6569	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000001001 000000111 000000111 100000010 110000001 001100011 101110100 111101100	{56}
1215	9	3/4	20	{2, 3, 2, 3}	(0, 0)	0	000011000 000010111 000001111 000001111 110000011 101100010 011100001 011111000 011110100	{8}
1216	9	3/4	20	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011000 000010111 000001111 000001111 110000011 101100010 011100001 011111000 011110100	{8}
1217	9	3/4	2.97808	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100000 111011100	{4}
1218	9	3/4	2.97808	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100000 111011100	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1219	9	3/4	2.92099	{1, 2, 1, 2}	(0, 0)	0	000001011 000000111 000000111 000000010 000000010 100000001 011000001 111110000 111001100	{4}
1220	9	3/4	2.92099	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000111 000000111 000000010 000000010 100000001 011000001 111110000 111001100	{4}
1221	9	3/4	2.7388	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000011 110000000 101100001 011110000 101110100	{4}
1222	9	3/4	2.7388	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000011 110000000 101100001 011110000 101110100	{4}
1223	9	3/4	2.68629	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001001 000000111 000000111 111000011 110110010 110111101 111111010	{8}
1224	9	3/4	2.68629	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001001 000000111 000000111 111000011 110110010 110111101 111111010	{8}
1225	9	3/4	2.66667	{2, 3, 2, 3}	(0, 0)	0	000000110 000000101 000000011 000000011 000000001 000000001 110000000 101100000 011111000	{4}
1226	9	3/4	2.66667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000110 000000101 000000011 000000011 000000001 000000001 110000000 101100000 011111000	{4}
1227	9	3/4	2.47944	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000000110 100000001 011000001 111100000 111100000 111011000	{4}
1228	9	3/4	2.47944	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000000110 100000001 011000001 111100000 111100000 111011000	{4}
1229	9	3/4	2.4	{2, 3, 2, 3}	(0, 0)	0	000001100 000001011 000000111 000000111 000000011 110000000 101100000 011110000 011110000	{4}
1230	9	3/4	2.4	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000111 000000011 110000000 101100000 011110000 011110000	{4}
1231	9	3/4	2	{0, 4, 5, 3}	(0, 0)	2.35619	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
1232	9	3/4	2	{4, 0, 3, 5}	(0, 0)	-2.35619	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
1233	9	3/4	2	{4, 0, 5, 3}	(1.5708, 1.1781)	-3.14159	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
1234	9	3/4	2	{0, 4, 3, 5}	(1.5708, -1.1781)	3.14159	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
1235	9	3/4	18.4853	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000011 101000001 110111001 101111110	{24}
1236	9	3/4	18.4853	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000011 101000001 110111001 101111110	{24}
1237	9	3/4	18.3431	{6, 7, 6, 7}	(0, 0)	0	000101111 000010111 000010110 100001111 011001111 100110111 111111001 111111001 110111110	{4}
1238	9	3/4	18.3431	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000101111 000010111 000010110 100001111 011001111 100110111 111111001 111111001 110111110	{4}
1239	9	3/4	17.6569	{3, 4, 3, 4}	(0, 0)	0	000001010 000001001 000000110 000000101 000000101 110000011 001110010 101001101 010111010	{24}
1240	9	3/4	17.6569	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001010 000001001 000000110 000000101 000000101 110000011 001110010 101001101 010111010	{24}
1241	9	3/4	17.4853	{3, 4, 3, 4}	(0, 0)	0	000001011 000000110 000000101 000000011 000000011 100000001 011000011 110110100 101111100	{4}
1242	9	3/4	17.4853	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001011 000000110 000000101 000000011 000000011 100000001 011000011 110110100 101111100	{4}
1243	9	3/4	16	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001011 000000101 110000110 111000000 110110000 111010001 111100010	{16}
1244	9	3/4	16	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001011 000000101 110000110 111000000 110110000 111010001 111100010	{16}
1245	9	3/4	15.9629	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000001 110000111 101101001 011101000 101111100	{4}
1246	9	3/4	15.9629	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000001 110000111 101101001 011101000 101111100	{4}
1247	9	3/4	126.225	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000000110 110000001 111000111 111101011 111101101 111011110	{12}
1248	9	3/4	126.225	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000000110 110000001 111000111 111101011 111101101 111011110	{12}
1249	9	3/4	12.5778	{0, 1, 0, 1}	(0, 0)	0	000011101 000011101 000011011 000000110 111000011 111000011 110100010 001111101 111011010	{4}
1250	9	3/4	12.5778	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011101 000011101 000011011 000000110 111000011 111000011 110100010 001111101 111011010	{4}
1251	9	3/4	12.3235	{1, 2, 1, 2}	(0, 0)	0	000001101 000000111 000000111 000000011 000000011 100000001 111000000 011110001 111111010	{4}
1252	9	3/4	12.3235	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001101 000000111 000000111 000000011 000000011 100000001 111000000 011110001 111111010	{4}
1253	9	3/4	116.569	{2, 3, 2, 3}	(0, 0)	0	000011110 000010001 000001111 000001111 110000011 101100010 101100001 101111000 011110100	{8}
1254	9	3/4	116.569	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000010001 000001111 000001111 110000011 101100010 101100001 101111000 011110100	{8}
1255	9	3/4	102.912	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000011 100000101 111000010 011010000 111101001 111110010	{8}
1256	9	3/4	102.912	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000011 100000101 111000010 011010000 111101001 111110010	{8}
1257	9	3/4	10.3431	{2, 3, 2, 3}	(0, 0)	0	000010111 000001101 000001011 000001011 100000110 011100111 110011001 101111000 111101100	{36}
1258	9	3/4	10.3431	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001101 000001011 000001011 100000110 011100111 110011001 101111000 111101100	{36}
1259	9	3/4	1.91912	{0, 7, 1, 6}	(1.5708, -1.5708)	-1.23096	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
1260	9	3/4	1.91912	{7, 0, 6, 1}	(1.5708, 1.5708)	-1.23096	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
1261	9	3/4	1.91912	{0, 7, 6, 1}	(0.955317, 0)	-3.14159	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
1262	9	3/4	1.91912	{7, 0, 1, 6}	(0.955317, -3.14159)	-3.14159	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
1263	9	3/4	1.89736	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000011 100000111 111000110 011011001 111111000 111110100	{4}
1264	9	3/4	1.89736	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000011 100000111 111000110 011011001 111111000 111110100	{4}
1265	9	3/4	1.84198	{4, 5, 4, 5}	(0, 0)	0	000000101 000000100 000000010 000000010 000000001 000000001 110000001 001100001 100011110	{4}
1266	9	3/4	1.84198	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000100 000000010 000000010 000000001 000000001 110000001 001100001 100011110	{4}
1267	9	3/4	1.74452	{1, 2, 1, 2}	(0, 0)	0	000001001 000000101 000000101 000000010 000000010 100000001 011000011 000110101 111001110	{4}
1268	9	3/4	1.74452	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001001 000000101 000000101 000000010 000000010 100000001 011000011 000110101 111001110	{4}
1269	9	3/4	1.70711	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000010 000000010 000000001 000000001 110000001 001100001 110011110	{4}
1270	9	3/4	1.70711	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000010 000000010 000000001 000000001 110000001 001100001 110011110	{4}
1271	9	3/4	1.37258	{2, 3, 2, 3}	(0, 0)	0	000010110 000001111 000000001 000000001 100000101 010000010 110010000 110001000 011110000	{8}
1272	9	3/4	1.37258	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010110 000001111 000000001 000000001 100000101 010000010 110010000 110001000 011110000	{8}
1273	9	3/4	1.33333	{4, 5, 4, 5}	(0, 0)	0	000000100 000000100 000000011 000000010 000000001 000000001 110000001 001100000 001011100	{4}
1274	9	3/4	1.33333	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000100 000000100 000000011 000000010 000000001 000000001 110000001 001100000 001011100	{4}
1275	9	3/4	0.958871	{4, 5, 4, 5}	(0, 0)	0	000000101 000000100 000000011 000000011 000000001 000000001 110000001 001100000 101111100	{4}
1276	9	3/4	0.958871	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000100 000000011 000000011 000000001 000000001 110000001 001100000 101111100	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1277	9	3/4	0.931773	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000111 000000011 000000011 100000001 011000000 011110001 111111010	{4}
1278	9	3/4	0.931773	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000111 000000011 000000011 100000001 011000000 011110001 111111010	{4}
1279	9	3/4	0.920991	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100000 111111100	{4}
1280	9	3/4	0.920991	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100000 111111100	{4}
1281	9	3/4	0.8	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000000 001100000 111011000	{4}
1282	9	3/4	0.8	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000000 001100000 111011000	{4}
1283	9	3/4	0.686292	{2, 3, 2, 3}	(0, 0)	0	000010111 000001110 000000001 000000001 100000100 010000011 110010001 110001000 101101100	{4}
1284	9	3/4	0.686292	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001110 000000001 000000001 100000100 010000011 110010001 110001000 101101100	{4}
1285	9	3/4	0.666667	{4, 5, 4, 5}	(0, 0)	0	000000100 000000100 000000011 000000011 000000001 000000001 110000001 001100000 001111100	{4}
1286	9	3/4	0.666667	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000100 000000100 000000011 000000011 000000001 000000001 110000001 001100000 001111100	{4}
1287	9	3/4	0.5	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000000 001100000 111111000	{4}
1288	9	3/4	0.5	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000000 001100000 111111000	{4}
1289	9	2/3	9.5	{5, 0, 3, 0}	(0, 0)	-3.14159	000101101 000010111 000001101 100000011 010000111 101000011 111010001 010111000 111111100	{4}
1290	9	2/3	9.5	{0, 5, 3, 0}	(1.5708, -1.5708)	3.14159	000101101 000010111 000001101 100000011 010000111 101000011 111010001 010111000 111111100	{4}
1291	9	2/3	9.33333	{1, 2, 1, 2}	(0, 0)	0	000110011 000001101 000001101 100010011 100100011 011000111 011001010 100111100 111111000	{8}
1292	9	2/3	9.33333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000110011 000001101 000001101 100010011 100100011 011000111 011001010 100111100 111111000	{8}
1293	9	2/3	9	{0, 4, 1, 3}	(0, 0)	2.0944	000100110 000010111 000001001 100000101 010000011 001000000 110100011 110010100 011110100	{16}
1294	9	2/3	9	{7, 3, 5, 8}	(0, 0)	1.0472	000001101 000001011 000000111 000000001 000000001 110000010 101000000 011001000 111110000	{24}
1295	9	2/3	9	{4, 0, 3, 1}	(0, 0)	-2.0944	000100110 000010111 000001001 100000101 010000011 001000000 110100011 110010100 011110100	{16}
1296	9	2/3	9	{3, 7, 8, 5}	(0, 0)	-1.0472	000001101 000001011 000000111 000000001 000000001 110000010 101000000 011001000 111110000	{24}
1297	9	2/3	9	{4, 0, 1, 3}	(1.5708, 1.0472)	-3.14159	000100110 000010111 000001001 100000101 010000011 001000000 110100011 110010100 011110100	{16}
1298	9	2/3	9	{3, 7, 5, 8}	(1.5708, 0.523599)	-3.14159	000001101 000001011 000000111 000000001 000000001 110000010 101000000 011001000 111110000	{24}
1299	9	2/3	9	{0, 4, 3, 1}	(1.5708, -1.0472)	3.14159	000100110 000010111 000001001 100000101 010000011 001000000 110100011 110010100 011110100	{16}
1300	9	2/3	9	{7, 3, 8, 5}	(1.5708, -0.523599)	3.14159	000001101 000001011 000000111 000000001 000000001 110000010 101000000 011001000 111110000	{24}
1301	9	2/3	8.5	{4, 8, 7, 5}	(0, 0)	0	000100111 000011100 000011011 100000111 011001010 011010001 110100000 101110001 101101010	{8}
1302	9	2/3	8.5	{4, 8, 5, 7}	(1.5708, 0)	-3.14159	000100111 000011100 000011011 100000111 011001010 011010001 110100000 101110001 101101010	{8}
1303	9	2/3	8	{3, 1, 3, 5}	(0, 0)	2.0944	000001110 000001011 000000101 000000011 000000001 110000011 101000001 110101000 011111100	{182}
1304	9	2/3	8	{0, 7, 8, 7}	(0, 0)	-3.14159	000010111 000001111 000001011 000000111 100000011 011000000 110100001 111110000 111110100	{73}
1305	9	2/3	8	{1, 3, 5, 3}	(0, 0)	-2.0944	000001110 000001011 000000101 000000011 000000001 110000011 101000001 110101000 011111100	{182}
1306	9	2/3	8	{1, 3, 3, 5}	(1.5708, 1.0472)	-3.14159	000001110 000001011 000000101 000000011 000000001 110000011 101000001 110101000 011111100	{182}
1307	9	2/3	8	{7, 0, 8, 7}	(1.5708, -1.5708)	3.14159	000010111 000001111 000001011 000000111 100000011 011000000 110100001 111110000 111110100	{73}
1308	9	2/3	8	{3, 1, 5, 3}	(1.5708, -1.0472)	3.14159	000001110 000001011 000000101 000000011 000000001 110000011 101000001 110101000 011111100	{182}
1309	9	2/3	7.5	{8, 6, 8, 7}	(0, 0)	1.0472	000010011 000001101 000001011 000000101 100000011 011000000 010100000 101010000 111110000	{56}
1310	9	2/3	7.5	{6, 8, 7, 8}	(0, 0)	-1.0472	000010011 000001101 000001011 000000101 100000011 011000000 010100000 101010000 111110000	{56}
1311	9	2/3	7.5	{6, 8, 8, 7}	(1.5708, 0.523599)	-3.14159	000010011 000001101 000001011 000000101 100000011 011000000 010100000 101010000 111110000	{56}
1312	9	2/3	7.5	{8, 6, 7, 8}	(1.5708, -0.523599)	3.14159	000010011 000001101 000001011 000000101 100000011 011000000 010100000 101010000 111110000	{56}
1313	9	2/3	7.33333	{7, 8, 7, 8}	(0, 0)	0	000011011 000000111 000000111 000000011 100001011 100010011 011000000 111111000 111111000	{24}
1314	9	2/3	7.33333	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000011011 000000111 000000111 000000011 100001011 100010011 011000000 111111000 111111000	{24}
1315	9	2/3	7	{1, 5, 8, 4}	(0, 0)	2.0944	000010111 000001111 000001101 000000001 100000011 011000000 111000011 110010100 111110100	{4}
1316	9	2/3	7	{7, 1, 4, 3}	(0, 0)	1.0472	000001110 000001011 000000101 000000011 000000010 110000000 101000001 110110000 011100100	{56}
1317	9	2/3	7	{6, 0, 4, 7}	(0, 0)	-3.14159	000001111 000001011 000000110 000000010 000000001 110000001 101000001 111100000 110011100	{52}
1318	9	2/3	7	{5, 1, 4, 8}	(0, 0)	-2.0944	000010111 000001111 000001101 000000001 100000011 011000000 111000011 110010100 111110100	{4}
1319	9	2/3	7	{1, 7, 3, 4}	(0, 0)	-1.0472	000001110 000001011 000000101 000000011 000000010 110000000 101000001 110110000 011100100	{56}
1320	9	2/3	7	{5, 1, 8, 4}	(1.5708, 1.0472)	-3.14159	000010111 000001111 000001101 000000001 100000011 011000000 111000011 110010100 111110100	{4}
1321	9	2/3	7	{1, 7, 4, 3}	(1.5708, 0.523599)	-3.14159	000001110 000001011 000000101 000000011 000000010 110000000 101000001 110110000 011100100	{56}
1322	9	2/3	7	{0, 6, 4, 7}	(1.5708, -1.5708)	3.14159	000001111 000001011 000000110 000000010 000000001 110000001 101000001 111100000 110011100	{52}
1323	9	2/3	7	{1, 5, 4, 8}	(1.5708, -1.0472)	3.14159	000010111 000001111 000001101 000000001 100000011 011000000 111000011 110010100 111110100	{4}
1324	9	2/3	7	{7, 1, 3, 4}	(1.5708, -0.523599)	3.14159	000001110 000001011 000000101 000000011 000000010 110000000 101000001 110110000 011100100	{56}
1325	9	2/3	6.5	{0, 4, 1, 2}	(0, 0)	0	000010110 000001110 000001001 000000111 100000001 011000000 110100011 110100101 001110110	{32}
1326	9	2/3	6.5	{0, 4, 2, 1}	(1.5708, 0)	-3.14159	000010110 000001110 000001001 000000111 100000001 011000000 110100011 110100101 001110110	{32}
1327	9	2/3	6.25	{0, 1, 7, 6}	(1.5708, 3.14159)	1.0472	000011101 000011011 000001111 000000111 110000111 111000001 101110000 011110000 111111000	{4}
1328	9	2/3	6.25	{0, 1, 6, 7}	(1.5708, 3.14159)	-2.0944	000011101 000011011 000001111 000000111 110000111 111000001 101110000 011110000 111111000	{4}
1329	9	2/3	6	{0, 2, 1, 2}	(0, 0)	-3.14159	000001010 000000110 000000101 000000010 000000001 100000001 011000000 110100000 001011000	{408}
1330	9	2/3	6	{2, 0, 1, 2}	(1.5708, -1.5708)	3.14159	000001010 000000110 000000101 000000010 000000001 100000001 011000000 110100000 001011000	{408}
1331	9	2/3	5.75	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000001 110000001 101100001 011100000 101111100	{24}
1332	9	2/3	5.75	{2, 3, 3, 2}	(1.5708, 0)	3.14159	000001101 000001010 000000111 000000111 000000001 110000001 101100001 011100000 101111100	{24}
1333	9	2/3	5.5	{3, 2, 3, 5}	(0, 0)	2.0944	000101111 000010111 000001001 100000001 010000110 101000001 110010010 110010100 111101000	{49}
1334	9	2/3	5.5	{5, 1, 5, 6}	(0, 0)	1.0472	001010111 000101101 100010111 010001011 101000101 010100010 111010001 101101000 111110100	{3}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1335	9	2/3	5.5	{2, 3, 5, 3}	(0, 0)	-2.0944	000101111 000010111 000001001 100000001 010000110 101000001 110010010 110010100 111101000	{49}
1336	9	2/3	5.5	{1, 5, 6, 5}	(0, 0)	-1.0472	001010111 000101101 100010111 010001011 101000101 010100010 111010001 101101000 111110100	{3}
1337	9	2/3	5.5	{2, 3, 3, 5}	(1.5708, 1.0472)	-3.14159	000101111 000010111 000001001 100000001 010000110 101000001 110010010 110010100 111101000	{49}
1338	9	2/3	5.5	{1, 5, 5, 6}	(1.5708, 0.523599)	-3.14159	001010111 000101101 100010111 010001011 101000101 010100010 111010001 101101000 111110100	{3}
1339	9	2/3	5.5	{3, 2, 5, 3}	(1.5708, -1.0472)	3.14159	000101111 000010111 000001001 100000001 010000110 101000001 110010010 110010100 111101000	{49}
1340	9	2/3	5.5	{5, 1, 6, 5}	(1.5708, -0.523599)	3.14159	001010111 000101101 100010111 010001011 101000101 010100010 111010001 101101000 111110100	{3}
1341	9	2/3	5.33333	{1, 2, 1, 2}	(0, 0)	0	000110011 000001111 000001111 100010011 100100011 011000000 011000000 111110000 111110000	{24}
1342	9	2/3	5.33333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000110011 000001111 000001111 100010011 100100011 011000000 011000000 111110000 111110000	{24}
1343	9	2/3	5	{8, 2, 3, 5}	(0, 0)	2.0944	000001101 000000111 000000011 000000001 000000001 100000000 110000000 011000000 111110000	{1164}
1344	9	2/3	5	{4, 0, 2, 7}	(0, 0)	-3.14159	000001111 000001011 000000111 000000010 000000001 110000001 101000000 111100000 111011000	{113}
1345	9	2/3	5	{2, 8, 5, 3}	(0, 0)	-2.0944	000001101 000000111 000000011 000000001 000000001 100000000 110000000 011000000 111110000	{1164}
1346	9	2/3	5	{2, 8, 3, 5}	(1.5708, 1.0472)	-3.14159	000001101 000000111 000000011 000000001 000000001 100000000 110000000 011000000 111110000	{1164}
1347	9	2/3	5	{0, 4, 2, 7}	(1.5708, -1.5708)	3.14159	000001111 000001011 000000111 000000010 000000001 110000001 101000000 111100000 111011000	{113}
1348	9	2/3	5	{8, 2, 5, 3}	(1.5708, -1.0472)	3.14159	000001101 000000111 000000011 000000001 000000001 100000000 110000000 011000000 111110000	{1164}
1349	9	2/3	42	{1, 2, 1, 2}	(0, 0)	0	000111001 000011111 000011111 100000110 111000101 111000011 011110011 011101100 111011100	{4}
1350	9	2/3	42	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000111001 000011111 000011111 100000110 111000101 111000011 011110011 011101100 111011100	{4}
1351	9	2/3	40	{2, 3, 2, 3}	(0, 0)	0	000011001 000010101 000001111 000001111 110000110 101100011 011110011 001111100 111101100	{8}
1352	9	2/3	40	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011001 000010101 000001111 000001111 110000110 101100011 011110011 001111100 111101100	{8}
1353	9	2/3	4.75	{0, 1, 7, 6}	(1.5708, 3.14159)	1.0472	000011100 000011010 000001111 000000111 110000111 111000001 101110001 011110001 001111110	{4}
1354	9	2/3	4.75	{0, 1, 6, 7}	(1.5708, 3.14159)	-2.0944	000011100 000011010 000001111 000000111 110000111 111000001 101110001 011110001 001111110	{4}
1355	9	2/3	4.5	{0, 7, 5, 3}	(0, 0)	3.14159	000010010 000001010 000000101 000000010 100000001 010000001 001000001 110100001 001011110	{152}
1356	9	2/3	4.5	{1, 6, 8, 5}	(0, 0)	2.0944	001011111 000101111 100011111 010000010 101000011 111000111 111001001 111111001 111011110	{8}
1357	9	2/3	4.5	{8, 1, 2, 6}	(0, 0)	1.0472	000010011 000001101 000001011 000000101 100000011 011000000 010100001 101010000 111110100	{68}
1358	9	2/3	4.5	{2, 7, 6, 3}	(0, 0)	0	000001001 000000110 000000100 000000010 000000001 100000001 011000001 010100001 100011110	{412}
1359	9	2/3	4.5	{6, 1, 5, 8}	(0, 0)	-2.0944	001011111 000101111 100011111 010000010 101000011 111000111 111001001 111111001 111011110	{8}
1360	9	2/3	4.5	{1, 8, 6, 2}	(0, 0)	-1.0472	000010011 000001101 000001011 000000101 100000011 011000000 010100001 101010000 111110100	{68}
1361	9	2/3	4.5	{3, 6, 4, 0}	(0.588003, -2.0944)	-2.24593	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1362	9	2/3	4.5	{6, 3, 0, 4}	(0.588003, -1.0472)	2.24593	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1363	9	2/3	4.5	{4, 0, 3, 6}	(0.588003, 2.0944)	-2.24593	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1364	9	2/3	4.5	{0, 4, 6, 3}	(0.588003, 1.0472)	2.24593	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1365	9	2/3	4.5	{4, 0, 6, 3}	(1.10715, -2.0944)	2.63623	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1366	9	2/3	4.5	{0, 4, 3, 6}	(1.10715, -1.0472)	-2.63623	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1367	9	2/3	4.5	{6, 3, 4, 0}	(1.10715, 2.0944)	2.63623	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1368	9	2/3	4.5	{3, 6, 0, 4}	(1.10715, 1.0472)	-2.63623	000011100 000001111 000000111 000000011 100001011 110010000 111000001 011110000 011110100	{1}
1369	9	2/3	4.5	{0, 7, 3, 5}	(1.5708, 1.5708)	-3.14159	000010010 000001010 000000101 000000010 100000001 010000001 001000001 110100001 001011110	{152}
1370	9	2/3	4.5	{6, 1, 8, 5}	(1.5708, 1.0472)	-3.14159	001011111 000101111 100011111 010000010 101000011 111000111 111001001 111111001 111011110	{8}
1371	9	2/3	4.5	{1, 8, 2, 6}	(1.5708, 0.523599)	-3.14159	000010011 000001101 000001011 000000101 100000011 011000000 010100001 101010000 111110100	{68}
1372	9	2/3	4.5	{2, 7, 3, 6}	(1.5708, 0)	3.14159	000001001 000000110 000000100 000000010 000000001 100000001 011000001 010100001 100011110	{412}
1373	9	2/3	4.5	{1, 6, 5, 8}	(1.5708, -1.0472)	3.14159	001011111 000101111 100011111 010000010 101000011 111000111 111001001 111111001 111011110	{8}
1374	9	2/3	4.5	{8, 1, 6, 2}	(1.5708, -0.523599)	3.14159	000010011 000001101 000001011 000000101 100000011 011000000 010100001 101010000 111110100	{68}
1375	9	2/3	4.25	{0, 7, 5, 3}	(0, 0)	0	000101101 000011011 000010100 100000111 011001010 110010011 101100000 010111001 110101010	{16}
1376	9	2/3	4.25	{0, 1, 3, 4}	(1.5708, 3.14159)	1.0472	000101111 000011111 000001001 100000111 010000111 111000111 110111000 110111000 111111000	{4}
1377	9	2/3	4.25	{0, 1, 4, 3}	(1.5708, 3.14159)	-2.0944	000101111 000011111 000001001 100000111 010000111 111000111 110111000 110111000 111111000	{4}
1378	9	2/3	4.25	{0, 7, 3, 5}	(1.5708, 0)	-3.14159	000101101 000011011 000010100 100000111 011001010 110010011 101100000 010111001 110101010	{16}
1379	9	2/3	34	{1, 2, 1, 2}	(0, 0)	0	000011010 000001111 000001111 000000011 100000101 111000001 011010000 111100001 011111010	{20}
1380	9	2/3	34	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011010 000001111 000001111 000000011 100000101 111000001 011010000 111100001 011111010	{20}
1381	9	2/3	32	{7, 8, 7, 8}	(0, 0)	0	000010111 000001011 000001011 000000111 100000011 011000100 100101000 111110000 111110000	{16}
1382	9	2/3	32	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000010111 000001011 000001011 000000111 100000011 011000100 100101000 111110000 111110000	{16}
1383	9	2/3	3.83333	{1, 2, 1, 2}	(0, 0)	0	000010110 000001111 000001111 000000011 100000011 011000001 111000000 111110001 011111010	{20}
1384	9	2/3	3.83333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010110 000001111 000001111 000000011 100000011 011000001 111000000 111110001 011111010	{20}
1385	9	2/3	3.77778	{1, 2, 1, 2}	(0, 0)	0	000011010 000001111 000001111 000000011 100000101 111000101 011011000 111100000 011111000	{4}
1386	9	2/3	3.77778	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011010 000001111 000001111 000000011 100000101 111000101 011011000 111100000 011111000	{4}
1387	9	2/3	3.75	{1, 4, 8, 4}	(0, 0)	-3.14159	000100110 000010100 000001011 100000100 010000011 001000001 110100001 101010000 001011100	{9}
1388	9	2/3	3.75	{0, 1, 6, 5}	(1.5708, 3.14159)	1.0472	000011010 000010110 000001111 000001111 110001111 101110001 011110001 111110001 001111110	{4}
1389	9	2/3	3.75	{0, 1, 5, 6}	(1.5708, 3.14159)	-2.0944	000011010 000010110 000001111 000001111 110001111 101110001 011110001 111110001 001111110	{4}
1390	9	2/3	3.75	{4, 1, 8, 4}	(1.5708, -1.5708)	3.14159	000100110 000010100 000001011 100000100 010000011 001000001 110100001 101010000 001011100	{9}
1391	9	2/3	3.71429	{1, 2, 1, 2}	(0, 0)	0	000101111 000010011 000010011 100001111 011001101 100110111 100111000 111101001 111111010	{8}
1392	9	2/3	3.71429	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101111 000010011 000010011 100001111 011001101 100110111 100111000 111101001 111111010	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1393	9	2/3	3.5	{0, 4, 2, 3}	(0, 0)	1.0472	000001011 000000101 000000011 000000010 000000010 100000000 010000001 101110000 111000100	{153}
1394	9	2/3	3.5	{4, 0, 3, 2}	(0, 0)	-1.0472	000001011 000000101 000000011 000000010 000000010 100000000 010000001 101110000 111000100	{153}
1395	9	2/3	3.5	{0, 1, 8, 6}	(0.588003, -2.0944)	-2.24593	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1396	9	2/3	3.5	{1, 0, 6, 8}	(0.588003, -1.0472)	2.24593	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1397	9	2/3	3.5	{8, 6, 0, 1}	(0.588003, 2.0944)	-2.24593	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1398	9	2/3	3.5	{6, 8, 1, 0}	(0.588003, 1.0472)	2.24593	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1399	9	2/3	3.5	{8, 6, 1, 0}	(1.10715, -2.0944)	2.63623	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1400	9	2/3	3.5	{6, 8, 0, 1}	(1.10715, -1.0472)	-2.63623	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1401	9	2/3	3.5	{1, 0, 8, 6}	(1.10715, 2.0944)	2.63623	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1402	9	2/3	3.5	{0, 1, 6, 8}	(1.10715, 1.0472)	-2.63623	000011000 000001110 000000111 000000101 100000011 110000011 011100001 011011000 001111100	{1}
1403	9	2/3	3.5	{4, 0, 2, 3}	(1.5708, 0.523599)	-3.14159	000001011 000000101 000000011 000000010 000000010 100000000 010000001 101110000 111000100	{153}
1404	9	2/3	3.5	{0, 4, 3, 2}	(1.5708, -0.523599)	3.14159	000001011 000000101 000000011 000000010 000000010 100000000 010000001 101110000 111000100	{153}
1405	9	2/3	3.42857	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000101 100000011 011000000 011100000 111010000 111110000	{24}
1406	9	2/3	3.42857	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000101 100000011 011000000 011100000 111010000 111110000	{24}
1407	9	2/3	3.33333	{8, 1, 8, 2}	(0, 0)	2.0944	000011111 000000101 000000011 000000001 100001111 100010111 110011000 101011000 111111000	{6}
1408	9	2/3	3.33333	{1, 6, 2, 6}	(0, 0)	-3.14159	000101111 000010111 000000101 100001111 010000010 100100111 111101001 110111001 111101110	{11}
1409	9	2/3	3.33333	{1, 8, 2, 8}	(0, 0)	-2.0944	000011111 000000101 000000011 000000001 100001111 100010111 110011000 101011000 111111000	{6}
1410	9	2/3	3.33333	{1, 8, 8, 2}	(1.5708, 1.0472)	-3.14159	000011111 000000101 000000011 000000001 100001111 100010111 110011000 101011000 111111000	{6}
1411	9	2/3	3.33333	{6, 1, 2, 6}	(1.5708, -1.5708)	3.14159	000101111 000010111 000000101 100001111 010000010 100100111 111101001 110111001 111101110	{11}
1412	9	2/3	3.33333	{8, 1, 2, 8}	(1.5708, -1.0472)	3.14159	000011111 000000101 000000011 000000001 100001111 100010111 110011000 101011000 111111000	{6}
1413	9	2/3	3.25	{1, 2, 1, 2}	(0, 0)	0	000011011 000000101 000000101 000000001 100001011 100010011 011000000 100011001 111110010	{36}
1414	9	2/3	3.25	{0, 5, 4, 1}	(1.5708, 2.0944)	1.0472	000010111 000001111 000000111 000000001 100000100 010000011 111010001 111001000 111101100	{2}
1415	9	2/3	3.25	{6, 2, 4, 7}	(0.523599, -2.61799)	3.14159	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1416	9	2/3	3.25	{3, 8, 4, 1}	(0.857072, -1.5708)	-1.44547	000011110 000001111 000001000 000000111 111000001 110110001 110110001 010111110	{1}
1417	9	2/3	3.25	{0, 1, 5, 4}	(1.5708, 3.14159)	1.0472	000110110 000101110 000011101 110000101 101000011 011000011 111100011 110011101 001111110	{4}
1418	9	2/3	3.25	{5, 0, 1, 4}	(1.5708, -2.0944)	1.0472	000010111 000001111 000000111 000000001 100000100 010000011 111010001 111001000 111101100	{2}
1419	9	2/3	3.25	{8, 3, 1, 4}	(0.857072, -1.5708)	1.44547	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1420	9	2/3	3.25	{2, 6, 7, 4}	(0.523599, -0.523599)	3.14159	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1421	9	2/3	3.25	{4, 7, 6, 2}	(0.523599, 2.61799)	3.14159	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1422	9	2/3	3.25	{1, 4, 8, 3}	(0.857072, 1.5708)	1.44547	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1423	9	2/3	3.25	{0, 1, 5, 8}	(0.857072, -2.61799)	-1.44547	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1424	9	2/3	3.25	{0, 6, 7, 1}	(1.5708, -2.0944)	-1.0472	000101111 000011011 000001011 100000111 010000101 111000010 100110011 111101100 111110100	{4}
1425	9	2/3	3.25	{5, 8, 0, 1}	(0.857072, 2.61799)	-1.44547	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1426	9	2/3	3.25	{8, 5, 1, 0}	(0.857072, 0.523599)	1.44547	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1427	9	2/3	3.25	{4, 1, 3, 8}	(0.857072, 1.5708)	-1.44547	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1428	9	2/3	3.25	{7, 4, 2, 6}	(0.523599, 0.523599)	3.14159	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1429	9	2/3	3.25	{6, 0, 1, 7}	(1.5708, 2.0944)	-1.0472	000101111 000011011 000001011 100000111 010000101 111000010 100110011 111101100 111110100	{4}
1430	9	2/3	3.25	{1, 0, 8, 5}	(0.857072, -0.523599)	1.44547	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1431	9	2/3	3.25	{6, 0, 7, 1}	(1.10715, 3.14159)	2.63623	000101111 000011011 000001011 100000111 010000101 111000010 100110011 111101100 111110100	{4}
1432	9	2/3	3.25	{1, 0, 5, 8}	(1.28976, 2.61799)	2.24593	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1433	9	2/3	3.25	{4, 1, 8, 3}	(1.0472, -2.61799)	3.14159	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1434	9	2/3	3.25	{7, 4, 6, 2}	(1.28976, -1.5708)	-2.24593	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1435	9	2/3	3.25	{5, 8, 1, 0}	(1.28976, -2.61799)	2.24593	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1436	9	2/3	3.25	{8, 5, 0, 1}	(1.28976, -0.523599)	-2.24593	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1437	9	2/3	3.25	{0, 1, 8, 5}	(1.28976, 0.523599)	-2.24593	000011111 000010011 000001111 000000011 110000110 101000000 101010001 111110001 111100110	{1}
1438	9	2/3	3.25	{0, 6, 1, 7}	(1.10715, 0)	-2.63623	000101111 000011011 000001011 100000111 010000101 111000010 100110011 111101100 111110100	{4}
1439	9	2/3	3.25	{4, 7, 2, 6}	(1.28976, -1.5708)	2.24593	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1440	9	2/3	3.25	{1, 4, 3, 8}	(1.0472, -0.523599)	-3.14159	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1441	9	2/3	3.25	{8, 3, 4, 1}	(1.0472, 2.61799)	3.14159	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1442	9	2/3	3.25	{2, 6, 4, 7}	(1.28976, 1.5708)	2.24593	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1443	9	2/3	3.25	{5, 0, 4, 1}	(1.10715, -3.14159)	-2.63623	000010111 000001111 000000111 000000001 100000100 010000011 111010001 111001000 111101100	{2}
1444	9	2/3	3.25	{0, 1, 4, 5}	(1.5708, 3.14159)	-2.0944	000110110 000101110 000011101 110000101 101000011 011000011 111100011 110011101 001111110	{4}
1445	9	2/3	3.25	{6, 2, 7, 4}	(1.28976, 1.5708)	-2.24593	000011010 000010101 000001011 000000111 110000011 101000011 010100011 101111100 011111100	{2}
1446	9	2/3	3.25	{3, 8, 1, 4}	(1.0472, 0.523599)	-3.14159	000011110 000001111 000001000 000000111 100000111 111000001 110110001 110110001 010111110	{1}
1447	9	2/3	3.25	{0, 5, 1, 4}	(1.10715, 0)	2.63623	000010111 000001111 000000111 000000001 100000100 010000011 111010001 111001000 111101100	{2}
1448	9	2/3	3.25	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000000101 000000101 000000001 100001011 100010011 011000000 100011001 111111010	{36}
1449	9	2/3	3	{6, 0, 5, 0}	(0, 0)	-3.14159	000010111 000001110 000000011 000000001 100000100 010000001 110010001 111000000 101101100	{44}
1450	9	2/3	3	{5, 0, 1, 4}	(1.5708, 1.0472)	1.0472	000010011 000001011 000000111 000000111 100000010 010000001 001100000 111110000 111101000	{2}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1451	9	2/3	3	{0, 5, 4, 1}	(1.5708, -1.0472)	1.0472	000010011 000001011 000000111 000000111 100000010 010000001 001100000 111110000 111101000	{2}
1452	9	2/3	3	{0, 5, 1, 4}	(1.10715, 3.14159)	2.63623	000010011 000001011 000000111 000000111 100000010 010000001 001100000 111110000 111101000	{2}
1453	9	2/3	3	{5, 0, 4, 1}	(1.10715, 0)	-2.63623	000010011 000001011 000000111 000000111 100000010 010000001 001100000 111110000 111101000	{2}
1454	9	2/3	3	{0, 6, 5, 0}	(1.5708, -1.5708)	3.14159	000010111 000001110 000000011 000000001 100000100 010000001 110010001 111000000 101101100	{44}
1455	9	2/3	28	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000101 000000010 110000100 001101000 111010000 111100000	{48}
1456	9	2/3	28	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000101 000000010 110000100 001101000 111010000 111100000	{48}
1457	9	2/3	26	{7, 8, 7, 8}	(0, 0)	0	000010100 000001111 000001111 000000011 100000011 011000011 111000000 011111000 011111000	{32}
1458	9	2/3	26	{7, 8, 8, 7}	(1.5708, 0)	-3.14159	000010100 000001111 000001111 000000011 100000011 011000011 111000000 011111000 011111000	{32}
1459	9	2/3	24	{2, 3, 2, 3}	(0, 0)	0	000011111 000001100 000000111 000000111 100000001 110000011 111100000 101101000 101111000	{80}
1460	9	2/3	24	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001100 000000111 000000111 100000001 110000011 111100000 101101000 101111000	{80}
1461	9	2/3	22	{1, 2, 1, 2}	(0, 0)	0	000001010 000000101 000000101 000000011 000000011 100000000 011000010 100110100 011110000	{92}
1462	9	2/3	22	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001010 000000101 000000101 000000011 000000011 100000000 011000010 100110100 011110000	{92}
1463	9	2/3	20	{7, 8, 7, 8}	(0, 0)	0	000001100 000001011 000000111 000000111 000000011 110000011 101100000 011111000 011111000	{136}
1464	9	2/3	20	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001100 000001011 000000111 000000111 000000011 110000011 101100000 011111000 011111000	{136}
1465	9	2/3	2.85714	{1, 2, 1, 2}	(0, 0)	0	000001001 000000111 000000111 000000011 000000010 100000001 011000000 011110000 111101000	{40}
1466	9	2/3	2.85714	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001001 000000111 000000111 000000011 000000010 100000001 011000000 011110000 111101000	{40}
1467	9	2/3	2.83333	{2, 3, 2, 3}	(0, 0)	0	000010101 000001001 000000011 000000011 100000101 010000001 100010001 001100000 111111100	{28}
1468	9	2/3	2.83333	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001001 000000011 000000011 100000101 010000001 100010001 001100000 111111100	{28}
1469	9	2/3	2.75	{0, 6, 5, 4}	(0, 0)	3.14159	000101101 000010111 000001010 100000101 010000011 101000010 110100001 011011000 110110100	{32}
1470	9	2/3	2.75	{0, 6, 5, 1}	(0, 0)	0	000101101 000010111 000001010 100000101 010000011 101000010 110100001 011011000 110110100	{72}
1471	9	2/3	2.75	{4, 2, 8, 5}	(0.857072, 2.61799)	1.44547	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1472	9	2/3	2.75	{0, 5, 4, 1}	(1.5708, 2.0944)	1.0472	000010111 000001111 000000101 000000011 100000100 010000011 111010011 110101100 111101100	{2}
1473	9	2/3	2.75	{5, 8, 0, 6}	(0.523599, -2.61799)	3.14159	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1474	9	2/3	2.75	{2, 7, 0, 3}	(0.588003, -2.0944)	-2.24593	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1475	9	2/3	2.75	{8, 5, 4, 2}	(0.857072, -2.61799)	1.44547	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1476	9	2/3	2.75	{2, 3, 6, 5}	(1.5708, 3.14159)	1.0472	000001111 000001101 000001010 000000110 000000001 111000001 110100001 101100001 110011110	{16}
1477	9	2/3	2.75	{5, 8, 2, 4}	(0.857072, -0.523599)	-1.44547	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1478	9	2/3	2.75	{4, 5, 7, 6}	(0.588003, -2.0944)	2.24593	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1479	9	2/3	2.75	{2, 6, 0, 1}	(0.523599, -1.5708)	-3.14159	000010100 000001110 000001011 000000001 100000011 011000101 110001011 011010101 001111110	{2}
1480	9	2/3	2.75	{5, 4, 6, 7}	(0.588003, -1.0472)	-2.24593	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1481	9	2/3	2.75	{5, 0, 1, 4}	(1.5708, -2.0944)	1.0472	000010111 000001111 000000101 000000011 100000100 010000011 111010011 110101100 111101100	{2}
1482	9	2/3	2.75	{2, 4, 5, 8}	(0.857072, 0.523599)	-1.44547	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1483	9	2/3	2.75	{7, 2, 3, 0}	(0.588003, -1.0472)	2.24593	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1484	9	2/3	2.75	{8, 5, 6, 0}	(0.523599, -0.523599)	3.14159	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1485	9	2/3	2.75	{0, 6, 5, 8}	(0.523599, 2.61799)	3.14159	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1486	9	2/3	2.75	{7, 6, 4, 5}	(0.588003, 2.0944)	2.24593	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1487	9	2/3	2.75	{2, 6, 7, 8}	(0.588003, 3.14159)	-2.24593	000001111 000001010 000001000 000000111 000000100 111000001 100110000 110100001 100101010	{2}
1488	9	2/3	2.75	{7, 3, 8, 6}	(0.857072, -2.61799)	-1.44547	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1489	9	2/3	2.75	{6, 1, 2, 5}	(1.5708, 1.0472)	1.0472	000001111 000001001 000000101 000000011 000000011 110000100 101001000 100110000 111110000	{10}
1490	9	2/3	2.75	{0, 3, 2, 7}	(0.588003, 2.0944)	-2.24593	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1491	9	2/3	2.75	{0, 1, 2, 6}	(0.523599, 1.5708)	-3.14159	000010100 000001110 000001011 000000001 100000011 011000101 110001011 011010101 001111110	{2}
1492	9	2/3	2.75	{3, 0, 7, 2}	(0.588003, 1.0472)	2.24593	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1493	9	2/3	2.75	{8, 6, 7, 3}	(0.857072, 2.61799)	-1.44547	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1494	9	2/3	2.75	{1, 3, 2, 4}	(1.5708, -3.14159)	-1.0472	000001111 000001011 000001001 000000111 000000101 111000001 100110001 110100001 111111110	{44}
1495	9	2/3	2.75	{6, 8, 3, 7}	(0.857072, 0.523599)	1.44547	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1496	9	2/3	2.75	{6, 7, 5, 4}	(0.588003, 1.0472)	-2.24593	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1497	9	2/3	2.75	{6, 0, 8, 5}	(0.523599, 0.523599)	3.14159	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1498	9	2/3	2.75	{1, 6, 5, 2}	(1.5708, -1.0472)	1.0472	000001111 000001001 000000101 000000011 000000011 110000100 101001000 100110000 111110000	{10}
1499	9	2/3	2.75	{3, 7, 6, 8}	(0.857072, -0.523599)	1.44547	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1500	9	2/3	2.75	{6, 2, 8, 7}	(0.588003, 0)	2.24593	000001111 000001010 000001000 000000111 000000100 111000001 100110000 110100001 100101010	{2}
1501	9	2/3	2.75	{1, 6, 2, 5}	(1.10715, 3.14159)	2.63623	000001111 000001001 000000101 000000011 000000011 110000100 101001000 100110000 111110000	{10}
1502	9	2/3	2.75	{3, 7, 8, 6}	(1.28976, 2.61799)	2.24593	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1503	9	2/3	2.75	{6, 2, 7, 8}	(1.5708, 2.0944)	2.0944	000001111 000001010 000001000 000000111 000000100 111000001 100110000 110100001 100101010	{2}
1504	9	2/3	2.75	{6, 7, 4, 5}	(1.10715, -2.0944)	-2.63623	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1505	9	2/3	2.75	{6, 0, 5, 8}	(1.28976, -1.5708)	-2.24593	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1506	9	2/3	2.75	{8, 6, 3, 7}	(1.28976, -2.61799)	2.24593	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1507	9	2/3	2.75	{1, 3, 4, 2}	(1.5708, -3.14159)	2.0944	000001111 000001011 000001001 000000111 000000101 111000001 100110001 110100001 111111110	{44}
1508	9	2/3	2.75	{6, 8, 7, 3}	(1.28976, -0.523599)	-2.24593	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1509	9	2/3	2.75	{0, 3, 7, 2}	(1.10715, -2.0944)	2.63623	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1510	9	2/3	2.75	{1, 5, 7, 0}	(1.0472, -1.5708)	-3.14159	000011111 000001101 000000111 000000011 100000011 110000111 111001001 101111001 111111110	{4}
1511	9	2/3	2.75	{3, 0, 2, 7}	(1.10715, -1.0472)	-2.63623	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1512	9	2/3	2.75	{2, 6, 8, 7}	(1.5708, -2.0944)	2.0944	000001111 000001010 000001000 000000111 000000100 111000001 100110000 110100001 100101010	{2}
1513	9	2/3	2.75	{7, 3, 6, 8}	(1.28976, 0.523599)	-2.24593	000011111 000011111 000001100 000000111 110000011 111000001 111100011 110110100 110111100	{2}
1514	9	2/3	2.75	{6, 1, 5, 2}	(1.10715, 0)	-2.63623	000001111 000001001 000000101 000000011 000000011 110000100 101001000 100110000 111110000	{10}
1515	9	2/3	2.75	{0, 6, 8, 5}	(1.28976, -1.5708)	2.24593	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1516	9	2/3	2.75	{7, 6, 5, 4}	(1.10715, -1.0472)	2.63623	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1517	9	2/3	2.75	{7, 2, 0, 3}	(1.10715, 2.0944)	2.63623	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1518	9	2/3	2.75	{8, 5, 0, 6}	(1.28976, 1.5708)	2.24593	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1519	9	2/3	2.75	{5, 0, 4, 1}	(1.10715, -3.14159)	-2.63623	000010111 000001111 000000101 000000011 100000100 010000011 111010011 110101100 111101100	{2}
1520	9	2/3	2.75	{2, 4, 8, 5}	(1.28976, -2.61799)	-2.24593	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1521	9	2/3	2.75	{4, 5, 6, 7}	(1.10715, 2.0944)	-2.63623	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1522	9	2/3	2.75	{0, 7, 5, 1}	(1.0472, 1.5708)	-3.14159	000011111 000001101 000000111 000000011 100000011 110000111 111001001 101111001 111111110	{4}
1523	9	2/3	2.75	{5, 4, 7, 6}	(1.10715, 1.0472)	2.63623	000010111 000001111 000001111 000001100 100000010 011100001 111100001 111010001 111001110	{6}
1524	9	2/3	2.75	{8, 5, 2, 4}	(1.28976, 2.61799)	-2.24593	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1525	9	2/3	2.75	{2, 3, 5, 6}	(1.5708, 3.14159)	-2.0944	000001111 000001101 000001010 000000110 000000001 111000001 110100001 101100001 110011110	{16}
1526	9	2/3	2.75	{5, 8, 4, 2}	(1.28976, 0.523599)	2.24593	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1527	9	2/3	2.75	{5, 8, 6, 0}	(1.28976, 1.5708)	-2.24593	000111111 000011111 000001111 100000011 110000101 111000011 111010000 111101001 111111010	{2}
1528	9	2/3	2.75	{2, 7, 3, 0}	(1.10715, 1.0472)	-2.63623	000001111 000001100 000001011 000000110 000000011 111000001 110100001 101110001 101011110	{1}
1529	9	2/3	2.75	{4, 2, 5, 8}	(1.28976, -0.523599)	2.24593	000010110 000001111 000001100 000000011 100000011 011000001 111000001 110110001 010111110	{3}
1530	9	2/3	2.75	{0, 5, 1, 4}	(1.10715, 0)	2.63623	000010111 000001111 000000101 000000011 100000100 010000011 111010011 110101100 111101100	{2}
1531	9	2/3	2.75	{0, 6, 4, 5}	(1.5708, 1.5708)	-3.14159	000101101 000010111 000001010 100000101 010000011 101000010 110100001 011011000 110110100	{32}
1532	9	2/3	2.75	{0, 6, 1, 5}	(1.5708, 0)	3.14159	000101101 000010111 000001010 100000101 010000011 101000010 110100001 011011000 110110100	{72}
1533	9	2/3	2.66667	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001001 000000101 000000011 111000000 110100000 110010000 111110000	{16}
1534	9	2/3	2.66667	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001001 000000101 000000011 111000000 110100000 110010000 111110000	{16}
1535	9	2/3	2.5	{7, 4, 6, 8}	(1.5708, -2.0944)	-1.0472	000011011 000010111 000001111 000000111 110001110 101010011 011110001 111111001 111101110	{8}
1536	9	2/3	2.5	{4, 7, 8, 6}	(1.5708, 2.0944)	-1.0472	000011011 000010111 000001111 000000111 110001110 101010011 011110001 111111001 111101110	{8}
1537	9	2/3	2.5	{4, 7, 6, 8}	(1.10715, 3.14159)	2.63623	000011011 000010111 000001111 000000111 110001110 101010011 011110001 111111001 111101110	{8}
1538	9	2/3	2.5	{7, 4, 8, 6}	(1.10715, 0)	-2.63623	000011011 000010111 000001111 000000111 110001110 101010011 011110001 111111001 111101110	{8}
1539	9	2/3	2.375	{1, 5, 3, 7}	(0.588003, 3.14159)	2.24593	000010011 000001101 000001100 000000111 100000011 011000001 011100011 100110100 110111100	{2}
1540	9	2/3	2.375	{5, 0, 6, 3}	(0.857072, -1.5708)	-1.44547	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1541	9	2/3	2.375	{5, 1, 7, 3}	(0.588003, 0)	-2.24593	000010011 000001101 000001100 000000111 100000011 011000001 011100011 100110100 110111100	{2}
1542	9	2/3	2.375	{0, 5, 3, 6}	(0.857072, -1.5708)	1.44547	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1543	9	2/3	2.375	{3, 6, 0, 5}	(0.857072, 1.5708)	1.44547	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1544	9	2/3	2.375	{3, 7, 6, 8}	(0.588003, 3.14159)	-2.24593	000010111 000001111 000001111 000001001 100000111 011100011 111010010 111011100 111111000	{4}
1545	9	2/3	2.375	{6, 3, 5, 0}	(0.857072, 1.5708)	-1.44547	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1546	9	2/3	2.375	{7, 3, 8, 6}	(0.588003, 0)	2.24593	000010111 000001111 000001111 000001001 100000111 011100011 111010010 111011100 111111000	{4}
1547	9	2/3	2.375	{7, 3, 6, 8}	(1.5708, 2.0944)	2.0944	000010111 000001111 000001111 000001001 100000111 011100011 111010010 111011100 111111000	{4}
1548	9	2/3	2.375	{6, 3, 0, 5}	(1.0472, -2.61799)	3.14159	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1549	9	2/3	2.375	{3, 7, 8, 6}	(1.5708, -2.0944)	2.0944	000010111 000001111 000001111 000001001 100000111 011100011 111010010 111011100 111111000	{4}
1550	9	2/3	2.375	{3, 6, 5, 0}	(1.0472, -0.523599)	-3.14159	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1551	9	2/3	2.375	{0, 5, 6, 3}	(1.0472, 2.61799)	3.14159	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1552	9	2/3	2.375	{5, 1, 3, 7}	(1.5708, -2.0944)	-2.0944	000010011 000001101 000001100 000000111 100000011 011000001 011100011 100110100 110111100	{2}
1553	9	2/3	2.375	{5, 0, 3, 6}	(1.0472, 0.523599)	-3.14159	000101111 000010111 000001101 100000011 010000111 101000011 111010011 110111100 111111100	{3}
1554	9	2/3	2.375	{1, 5, 7, 3}	(1.5708, 2.0944)	-2.0944	000010011 000001101 000001100 000000111 100000011 011000001 011100011 100110100 110111100	{2}
1555	9	2/3	2.33333	{1, 6, 7, 5}	(0, 0)	3.14159	001011111 000100111 100011111 010000101 101001111 101010011 111110011 111011101 111111110	{8}
1556	9	2/3	2.33333	{1, 6, 7, 2}	(0, 0)	1.0472	000101011 000010111 000000100 100001011 010000101 100100011 011010000 110101001 110111010	{12}
1557	9	2/3	2.33333	{6, 1, 2, 7}	(0, 0)	-1.0472	000101011 000010111 000000100 100001011 010000101 100100011 011010000 110101001 110111010	{12}
1558	9	2/3	2.33333	{1, 6, 5, 7}	(1.5708, 1.5708)	-3.14159	001011111 000100111 100011111 010000101 101001111 101010011 111110011 111011101 111111110	{8}
1559	9	2/3	2.33333	{6, 1, 7, 2}	(1.5708, 0.523599)	-3.14159	000101011 000010111 000000100 100001011 010000101 100100011 011010000 110101001 110111010	{12}
1560	9	2/3	2.33333	{1, 6, 2, 7}	(1.5708, -0.523599)	3.14159	000101011 000010111 000000100 100001011 010000101 100100011 011010000 110101001 110111010	{12}
1561	9	2/3	2.25	{1, 5, 8, 3}	(0, 0)	3.14159	000101011 000010101 000001111 100000011 010000100 101000011 011010000 101101001 111101010	{16}
1562	9	2/3	2.25	{0, 4, 1, 3}	(0, 0)	2.0944	000101111 000011111 000000011 100001010 010000101 110100111 110011011 111101101 111011110	{32}
1563	9	2/3	2.25	{4, 0, 3, 1}	(0, 0)	-2.0944	000101111 000011111 000000011 100001010 010000101 110100111 110011011 111101101 111011110	{32}
1564	9	2/3	2.25	{7, 1, 0, 5}	(0.857072, 2.61799)	1.44547	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1565	9	2/3	2.25	{0, 6, 5, 1}	(1.5708, 2.0944)	1.0472	000001011 000000111 000000011 000000010 000000001 100000010 010000001 111101001 111010110	{2}
1566	9	2/3	2.25	{2, 5, 6, 0}	(0.523599, -2.61799)	-3.14159	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{1}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1567	9	2/3	2.25	{2, 1, 4, 6}	(0.588003, -2.0944)	-2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1568	9	2/3	2.25	{5, 8, 7, 3}	(0.857072, -1.5708)	-1.44547	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1569	9	2/3	2.25	{0, 5, 7, 1}	(0.857072, -2.61799)	1.44547	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1570	9	2/3	2.25	{0, 1, 5, 6}	(1.5708, 3.14159)	1.0472	000001011 000000111 000000011 000000010 000000010 100000001 010000001 111110001 111001110	{12}
1571	9	2/3	2.25	{5, 0, 1, 7}	(0.857072, -0.523599)	-1.44547	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1572	9	2/3	2.25	{6, 8, 1, 0}	(0.588003, -2.0944)	2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1573	9	2/3	2.25	{0, 3, 5, 6}	(0.523599, -1.5708)	-3.14159	000001111 000000111 000000101 000000011 000000001 100000010 111000000 110101001 111110010	{2}
1574	9	2/3	2.25	{8, 6, 0, 1}	(0.588003, -1.0472)	-2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1575	9	2/3	2.25	{6, 0, 1, 5}	(1.5708, -2.0944)	1.0472	000001011 000000111 000000011 000000010 000000001 100000010 010000001 111101001 111010110	{2}
1576	9	2/3	2.25	{1, 7, 5, 0}	(0.857072, 0.523599)	-1.44547	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1577	9	2/3	2.25	{8, 5, 3, 7}	(0.857072, -1.5708)	1.44547	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1578	9	2/3	2.25	{1, 2, 6, 4}	(0.588003, -1.0472)	2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1579	9	2/3	2.25	{5, 2, 0, 6}	(0.523599, -0.523599)	-3.14159	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{1}
1580	9	2/3	2.25	{6, 0, 2, 5}	(0.523599, 2.61799)	-3.14159	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{1}
1581	9	2/3	2.25	{1, 0, 6, 8}	(0.588003, 2.0944)	2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1582	9	2/3	2.25	{3, 7, 8, 5}	(0.857072, 1.5708)	1.44547	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1583	9	2/3	2.25	{3, 5, 6, 4}	(0.588003, 3.14159)	-2.24593	000011000 000001101 000000111 000000011 100000010 110000000 011000000 001110000 011100000	{2}
1584	9	2/3	2.25	{5, 0, 7, 8}	(0.857072, -2.61799)	-1.44547	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1585	9	2/3	2.25	{5, 0, 6, 1}	(1.5708, -2.0944)	-1.0472	000011111 000001111 000000101 000000010 100000001 110000011 111000011 110101101 111011110	{6}
1586	9	2/3	2.25	{4, 6, 2, 1}	(0.588003, 2.0944)	-2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1587	9	2/3	2.25	{5, 6, 0, 3}	(0.523599, 1.5708)	-3.14159	000001111 000000111 000000011 000000010 000000001 100000010 111000000 110101001 111110010	{2}
1588	9	2/3	2.25	{6, 4, 1, 2}	(0.588003, 1.0472)	2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1589	9	2/3	2.25	{7, 8, 5, 0}	(0.857072, 2.61799)	-1.44547	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1590	9	2/3	2.25	{0, 6, 1, 5}	(1.5708, -3.14159)	-1.0472	000011001 000001101 000000110 000000011 100000010 110000001 011000001 001110001 110101110	{32}
1591	9	2/3	2.25	{8, 7, 0, 5}	(0.857072, 0.523599)	1.44547	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1592	9	2/3	2.25	{7, 3, 5, 8}	(0.857072, 1.5708)	-1.44547	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1593	9	2/3	2.25	{0, 1, 8, 6}	(0.588003, 1.0472)	-2.24593	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1594	9	2/3	2.25	{0, 6, 5, 2}	(0.523599, 0.523599)	-3.14159	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{1}
1595	9	2/3	2.25	{0, 5, 1, 6}	(1.5708, 2.0944)	-1.0472	000011111 000001111 000000101 000000010 100000001 110000011 111000011 110101101 111011110	{6}
1596	9	2/3	2.25	{0, 5, 8, 7}	(0.857072, -0.523599)	1.44547	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1597	9	2/3	2.25	{5, 3, 4, 6}	(0.588003, 0)	2.24593	000011000 000001101 000000111 000000011 100000010 110000000 011000000 001110000 011100000	{2}
1598	9	2/3	2.25	{0, 5, 6, 1}	(1.10715, 3.14159)	2.63623	000011111 000001111 000000101 000000010 100000001 110000011 111000011 110101101 111011110	{6}
1599	9	2/3	2.25	{0, 5, 7, 8}	(1.28976, 2.61799)	2.24593	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1600	9	2/3	2.25	{5, 3, 6, 4}	(1.5708, 2.0944)	2.0944	000011000 000001101 000000111 000000011 100000010 110000000 011000000 001110000 011100000	{2}
1601	9	2/3	2.25	{7, 3, 8, 5}	(1.0472, -2.61799)	3.14159	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1602	9	2/3	2.25	{0, 1, 6, 8}	(1.10715, -2.0944)	-2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1603	9	2/3	2.25	{0, 6, 2, 5}	(1.28976, -1.5708)	-2.24593	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{3}
1604	9	2/3	2.25	{7, 8, 0, 5}	(1.28976, -2.61799)	2.24593	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1605	9	2/3	2.25	{0, 6, 5, 1}	(1.5708, -3.14159)	2.0944	000011001 000001101 000000110 000000011 100000010 110000001 011000001 001110001 110101110	{32}
1606	9	2/3	2.25	{8, 7, 5, 0}	(1.28976, -0.523599)	-2.24593	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1607	9	2/3	2.25	{4, 6, 1, 2}	(1.10715, -2.0944)	2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1608	9	2/3	2.25	{5, 6, 3, 0}	(1.0472, -1.5708)	3.14159	000001111 000000111 000000101 000000011 000000001 100000010 111000000 110101001 111110010	{4}
1609	9	2/3	2.25	{6, 4, 2, 1}	(1.10715, -1.0472)	-2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1610	9	2/3	2.25	{3, 5, 4, 6}	(1.5708, -2.0944)	2.0944	000011000 000001101 000000111 000000011 100000010 110000000 011000000 001110000 011100000	{2}
1611	9	2/3	2.25	{5, 0, 8, 7}	(1.28976, 0.523599)	-2.24593	000001011 000001010 000000110 000000100 000000100 110000001 001110001 111000001 100001110	{2}
1612	9	2/3	2.25	{5, 0, 1, 6}	(1.10715, 0)	-2.63623	000011111 000001111 000000101 000000010 100000001 110000011 111000011 110101101 111011110	{6}
1613	9	2/3	2.25	{6, 0, 5, 2}	(1.28976, -1.5708)	2.24593	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{3}
1614	9	2/3	2.25	{1, 0, 8, 6}	(1.10715, -1.0472)	2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1615	9	2/3	2.25	{3, 7, 5, 8}	(1.0472, -0.523599)	-3.14159	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1616	9	2/3	2.25	{8, 5, 7, 3}	(1.0472, 2.61799)	3.14159	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1617	9	2/3	2.25	{1, 2, 4, 6}	(1.10715, 2.0944)	2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1618	9	2/3	2.25	{5, 2, 6, 0}	(1.28976, 1.5708)	2.24593	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{3}
1619	9	2/3	2.25	{6, 0, 5, 1}	(1.10715, -3.14159)	-2.63623	000001011 000000111 000000011 000000010 000000001 100000010 010000001 111101001 111010110	{2}
1620	9	2/3	2.25	{1, 7, 0, 5}	(1.28976, -2.61799)	-2.24593	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1621	9	2/3	2.25	{6, 8, 0, 1}	(1.10715, 2.0944)	-2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1622	9	2/3	2.25	{0, 3, 6, 5}	(1.0472, 1.5708)	3.14159	000001111 000000111 000000101 000000011 000000001 100000010 111000000 110101001 111110010	{4}
1623	9	2/3	2.25	{8, 6, 1, 0}	(1.10715, 1.0472)	2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{5}
1624	9	2/3	2.25	{0, 5, 1, 7}	(1.28976, 2.61799)	-2.24593	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1625	9	2/3	2.25	{0, 1, 6, 5}	(1.5708, 3.14159)	-2.0944	000001011 000000111 000000011 000000010 000000010 100000001 010000001 111110001 111001110	{12}
1626	9	2/3	2.25	{5, 0, 7, 1}	(1.28976, 0.523599)	2.24593	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1627	9	2/3	2.25	{2, 5, 0, 6}	(1.28976, 1.5708)	-2.24593	000001111 000000111 000000011 000000010 000000001 100000101 110001000 111100001 111011010	{3}
1628	9	2/3	2.25	{2, 1, 6, 4}	(1.10715, 1.0472)	-2.63623	000011010 000001100 000000111 000000010 100000011 110000001 011000001 101110001 001011110	{1}
1629	9	2/3	2.25	{5, 8, 3, 7}	(1.0472, 0.523599)	-3.14159	000001010 000001010 000000110 000000101 000000100 110000001 001110001 111000001 000101110	{2}
1630	9	2/3	2.25	{7, 1, 5, 0}	(1.28976, -0.523599)	2.24593	000110111 000011011 000001111 100000101 110000001 011000010 101100000 111001001 111110010	{1}
1631	9	2/3	2.25	{0, 6, 1, 5}	(1.10715, 0)	2.63623	000001011 000000111 000000011 000000010 000000001 100000010 010000001 111101001 111010110	{2}
1632	9	2/3	2.25	{1, 5, 3, 8}	(1.5708, 1.5708)	-3.14159	000101011 000010101 000001111 100000011 010000100 101000011 011010000 101101001 111101010	{16}
1633	9	2/3	2.25	{4, 0, 1, 3}	(1.5708, 1.0472)	-3.14159	000101111 000011111 000000011 100001010 010000101 110100111 110011011 111101101 111011110	{32}
1634	9	2/3	2.25	{0, 4, 3, 1}	(1.5708, -1.0472)	3.14159	000101111 000011111 000000011 100001010 010000101 110100111 110011011 111101101 111011110	{32}
1635	9	2/3	2.14286	{0, 3, 2, 1}	(0, 0)	0	000100101 000011100 000011010 100000011 011001111 011010111 110011011 001111101 100111110	{8}
1636	9	2/3	2.14286	{0, 3, 1, 2}	(1.5708, 0)	3.14159	000100101 000011100 000011010 100000011 011001111 011010111 110011011 001111101 100111110	{8}
1637	9	2/3	2.125	{6, 0, 7, 3}	(0.857072, -1.5708)	-1.44547	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1638	9	2/3	2.125	{0, 6, 3, 7}	(0.857072, -1.5708)	1.44547	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1639	9	2/3	2.125	{3, 7, 0, 6}	(0.857072, 1.5708)	1.44547	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1640	9	2/3	2.125	{3, 6, 4, 7}	(0.588003, 3.14159)	-2.24593	000011111 000011111 000010110 000001011 111000100 110100111 111011001 111101001 110101110	{2}
1641	9	2/3	2.125	{7, 3, 6, 0}	(0.857072, 1.5708)	-1.44547	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1642	9	2/3	2.125	{6, 3, 7, 4}	(0.588003, 0)	2.24593	000011111 000011111 000010110 000001011 111000100 110100111 111011001 111101001 110101110	{2}
1643	9	2/3	2.125	{6, 3, 4, 7}	(1.5708, 2.0944)	2.0944	000011111 000011111 000010110 000001011 111000100 110100111 111011001 111101001 110101110	{2}
1644	9	2/3	2.125	{7, 3, 0, 6}	(1.0472, -2.61799)	3.14159	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1645	9	2/3	2.125	{3, 6, 7, 4}	(1.5708, -2.0944)	2.0944	000011111 000011111 000010110 000001011 111000100 110100111 111011001 111101001 110101110	{2}
1646	9	2/3	2.125	{3, 7, 6, 0}	(1.0472, -0.523599)	-3.14159	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1647	9	2/3	2.125	{0, 6, 7, 3}	(1.0472, 2.61799)	3.14159	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1648	9	2/3	2.125	{6, 0, 3, 7}	(1.0472, 0.523599)	-3.14159	000101111 000010111 000001011 100001001 010000111 101100110 110011001 111011001 111110110	{1}
1649	9	2/3	2	{0, 3, 8, 2}	(0, 0)	3.14159	000010101 000001001 000000010 000000010 100000100 010000001 100010000 001100000 110001000	{16}
1650	9	2/3	2	{5, 1, 5, 8}	(0, 0)	1.0472	000101011 000010101 000000010 100001011 010000100 100100001 010010000 101100001 110101010	{7}
1651	9	2/3	2	{5, 2, 4, 3}	(0, 0)	0.380251	000011011 000011011 000000111 000000111 110000011 110000001 001100011 111110100 111111100	{16}
1652	9	2/3	2	{1, 5, 8, 5}	(0, 0)	-1.0472	000101011 000010101 000000010 100001011 010000100 100100001 010010000 101100001 110101010	{7}
1653	9	2/3	2	{2, 5, 3, 4}	(0, 0)	-0.380251	000011011 000011011 000000111 000000111 110000011 110000001 001100011 111110100 111111100	{16}
1654	9	2/3	2	{1, 0, 5, 3}	(0.785398, -2.0944)	2.3664	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1655	9	2/3	2	{0, 1, 3, 5}	(0.785398, -1.0472)	-2.3664	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1656	9	2/3	2	{5, 3, 1, 0}	(0.785398, 2.0944)	2.3664	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1657	9	2/3	2	{3, 5, 0, 1}	(0.785398, 1.0472)	-2.3664	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1658	9	2/3	2	{3, 5, 1, 0}	(0.927295, -2.0944)	-2.47465	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1659	9	2/3	2	{5, 3, 0, 1}	(0.927295, -1.0472)	2.47465	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1660	9	2/3	2	{1, 0, 3, 5}	(0.927295, 2.0944)	-2.47465	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1661	9	2/3	2	{0, 1, 5, 3}	(0.927295, 1.0472)	2.47465	000101011 000010110 000001111 100000010 010000101 101000001 011010000 111100001 101011010	{1}
1662	9	2/3	2	{0, 3, 2, 8}	(1.5708, 1.5708)	-3.14159	000010101 000001001 000000010 000000010 100000100 010000001 100010000 001100000 110001000	{16}
1663	9	2/3	2	{1, 5, 5, 8}	(1.5708, 0.523599)	-3.14159	000101011 000010101 000000010 100001011 010000100 100100001 010010000 101100001 110101010	{7}
1664	9	2/3	2	{2, 5, 4, 3}	(1.5708, 0.190126)	-3.14159	000011011 000011011 000000111 000000111 110000011 110000001 001100011 111110100 111111100	{16}
1665	9	2/3	2	{5, 1, 8, 5}	(1.5708, -0.523599)	3.14159	000101011 000010101 000000010 100001011 010000100 100100001 010010000 101100001 110101010	{7}
1666	9	2/3	2	{5, 2, 3, 4}	(1.5708, -0.190126)	3.14159	000011011 000011011 000000111 000000111 110000011 110000001 001100011 111110100 111111100	{16}
1667	9	2/3	19	{1, 4, 3, 2}	(0, 0)	0	000111101 000011110 000001011 100000011 110000110 111000101 110011001 011110001 101101110	{8}
1668	9	2/3	19	{1, 4, 2, 3}	(1.5708, 0)	3.14159	000111101 000011110 000001011 100000011 110000110 111000101 110011001 011110001 101101110	{8}
1669	9	2/3	18	{1, 2, 1, 2}	(0, 0)	0	000001101 000000111 000000111 000000001 000000001 100000010 111000000 011001000 111110000	{172}
1670	9	2/3	18	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001101 000000111 000000111 000000001 000000001 100000010 111000000 011001000 111110000	{172}
1671	9	2/3	17	{1, 8, 7, 5}	(0, 0)	0	000011110 000001111 000000011 000000011 100000111 110000011 110010001 111111000 011111100	{16}
1672	9	2/3	17	{1, 8, 5, 7}	(1.5708, 0)	3.14159	000011110 000001111 000000011 000000011 100000111 110000011 110010001 111111000 011111100	{16}
1673	9	2/3	16	{2, 3, 8, 3}	(0, 0)	-3.14159	000111100 000011111 000001011 100000111 110001111 111010001 110110010 011110101 011111010	{10}
1674	9	2/3	16	{3, 2, 8, 3}	(1.5708, -1.5708)	3.14159	000111100 000011111 000001011 100000111 110001111 111010001 110110010 011110101 011111010	{10}
1675	9	2/3	15.5	{2, 3, 2, 3}	(0, 0)	0	000010011 000001011 000000101 000000101 100000011 010000001 001100010 110010100 111111000	{28}
1676	9	2/3	15.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010011 000001011 000000101 000000101 100000011 010000001 001100010 110010100 111111000	{28}
1677	9	2/3	14	{1, 2, 1, 2}	(0, 0)	0	000011011 000000101 000000101 000000011 100001011 100010001 011000010 100110100 111111000	{256}
1678	9	2/3	14	{2, 3, 7, 3}	(0, 0)	-3.14159	000111001 000011110 000000111 100001011 110000101 110100001 011010010 011100101 101111010	{3}
1679	9	2/3	14	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000000101 000000101 000000011 100001011 100010001 011000010 100110100 111111000	{256}
1680	9	2/3	14	{3, 2, 7, 3}	(1.5708, -1.5708)	3.14159	000111001 000011110 000000111 100001011 110000101 110100001 011010010 011100101 101111010	{3}
1681	9	2/3	12	{2, 1, 2, 7}	(0, 0)	2.0944	000010100 000001111 000001101 000000011 100000011 011000011 111000010 010111101 011111010	{22}
1682	9	2/3	12	{1, 0, 1, 4}	(0, 0)	1.0472	000011010 000010111 000001101 000000011 110000100 101000011 011010001 110101000 011101100	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1683	9	2/3	12	{1, 2, 7, 2}	(0, 0)	-2.0944	000010100 000001111 000001101 000000011 100000011 011000011 111000010 010111101 011111010	{22}
1684	9	2/3	12	{0, 1, 4, 1}	(0, 0)	-1.0472	000011010 000010111 000001101 000000011 110000100 101000011 011010001 110101000 011101100	{8}
1685	9	2/3	12	{1, 2, 2, 7}	(1.5708, 1.0472)	-3.14159	000010100 000001111 000001101 000000011 100000011 011000011 111000010 010111101 011111010	{22}
1686	9	2/3	12	{0, 1, 1, 4}	(1.5708, 0.523599)	-3.14159	000011010 000010111 000001101 000000011 110000100 101000011 011010001 110101000 011101100	{8}
1687	9	2/3	12	{2, 1, 7, 2}	(1.5708, -1.0472)	3.14159	000010100 000001111 000001101 000000011 100000011 011000011 111000010 010111101 011111010	{22}
1688	9	2/3	12	{1, 0, 4, 1}	(1.5708, -0.523599)	3.14159	000011010 000010111 000001101 000000011 110000100 101000011 011010001 110101000 011101100	{8}
1689	9	2/3	11.5	{2, 3, 2, 3}	(0, 0)	0	000010010 000001001 000000111 000000111 100000001 010000001 001100001 101100000 011111100	{68}
1690	9	2/3	11.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010010 000001001 000000111 000000111 100000001 010000001 001100001 101100000 011111100	{68}
1691	9	2/3	11	{1, 8, 7, 5}	(0, 0)	0	000011010 000001111 000000111 000000011 100000011 110000011 011000001 111111000 011111100	{56}
1692	9	2/3	11	{1, 8, 5, 7}	(1.5708, 0)	3.14159	000011010 000001111 000000111 000000011 100000011 110000011 011000001 111111000 011111100	{56}
1693	9	2/3	10	{1, 3, 2, 3}	(0, 0)	2.0944	000010101 000001111 000001010 000000111 100000010 011000000 110100001 011110001 110100110	{38}
1694	9	2/3	10	{3, 7, 3, 4}	(0, 0)	1.0472	000101011 000010101 000001110 100000100 010000011 101000010 011100001 101011000 110010100	{15}
1695	9	2/3	10	{3, 1, 3, 2}	(0, 0)	-2.0944	000010101 000001111 000001010 000000111 100000010 011000000 110100001 011110001 110100110	{38}
1696	9	2/3	10	{7, 3, 4, 3}	(0, 0)	-1.0472	000101011 000010101 000001110 100000100 010000011 101000010 011100001 101011000 110010100	{15}
1697	9	2/3	10	{3, 1, 2, 3}	(1.5708, 1.0472)	-3.14159	000010101 000001111 000001010 000000111 100000010 011000000 110100001 011110001 110100110	{38}
1698	9	2/3	10	{7, 3, 3, 4}	(1.5708, 0.523599)	-3.14159	000101011 000010101 000001110 100000100 010000011 101000010 011100001 101011000 110010100	{15}
1699	9	2/3	10	{1, 3, 3, 2}	(1.5708, -1.0472)	3.14159	000010101 000001111 000001010 000000111 100000010 011000000 110100001 011110001 110100110	{38}
1700	9	2/3	10	{3, 7, 4, 3}	(1.5708, -0.523599)	3.14159	000101011 000010101 000001110 100000100 010000011 101000010 011100001 101011000 110010100	{15}
1701	9	2/3	1.875	{6, 5, 2, 7}	(0.588003, -2.0944)	-2.24593	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1702	9	2/3	1.875	{6, 0, 7, 4}	(0.857072, -1.5708)	-1.44547	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1703	9	2/3	1.875	{0, 6, 4, 7}	(0.857072, -1.5708)	1.44547	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1704	9	2/3	1.875	{5, 6, 7, 2}	(0.588003, -1.0472)	2.24593	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1705	9	2/3	1.875	{4, 7, 0, 6}	(0.857072, 1.5708)	1.44547	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1706	9	2/3	1.875	{2, 3, 5, 6}	(0.588003, 3.14159)	-2.24593	000010111 000001111 000001001 000000011 100000111 011000000 110010000 110110000 111110000	{8}
1707	9	2/3	1.875	{6, 4, 7, 0}	(0.857072, -2.61799)	-1.44547	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1708	9	2/3	1.875	{8, 5, 7, 6}	(1.5708, -2.0944)	-1.0472	000010011 000001101 000001100 000000111 100000011 011000000 011100000 100110000 110110000	{4}
1709	9	2/3	1.875	{2, 7, 6, 5}	(0.588003, 2.0944)	-2.24593	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1710	9	2/3	1.875	{7, 2, 5, 6}	(0.588003, 1.0472)	2.24593	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1711	9	2/3	1.875	{7, 0, 6, 4}	(0.857072, 2.61799)	-1.44547	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1712	9	2/3	1.875	{0, 7, 4, 6}	(0.857072, 0.523599)	1.44547	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1713	9	2/3	1.875	{7, 4, 6, 0}	(0.857072, 1.5708)	-1.44547	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1714	9	2/3	1.875	{5, 8, 6, 7}	(1.5708, 2.0944)	-1.0472	000010011 000001101 000001100 000000111 100000011 011000000 011100000 100110000 110110000	{4}
1715	9	2/3	1.875	{4, 6, 0, 7}	(0.857072, -0.523599)	1.44547	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1716	9	2/3	1.875	{3, 2, 6, 5}	(0.588003, 0)	2.24593	000010111 000001111 000001001 000000011 100000111 011000000 110010000 110110000 111110000	{8}
1717	9	2/3	1.875	{5, 8, 7, 6}	(1.10715, 3.14159)	2.63623	000010011 000001101 000001100 000000111 100000011 011000000 011100000 100110000 110110000	{4}
1718	9	2/3	1.875	{4, 6, 7, 0}	(1.28976, 2.61799)	2.24593	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1719	9	2/3	1.875	{3, 2, 5, 6}	(1.5708, 2.0944)	2.0944	000010111 000001111 000001001 000000011 100000111 011000000 110010000 110110000 111110000	{8}
1720	9	2/3	1.875	{7, 4, 0, 6}	(1.0472, -2.61799)	3.14159	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1721	9	2/3	1.875	{7, 0, 4, 6}	(1.28976, -2.61799)	2.24593	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1722	9	2/3	1.875	{0, 7, 6, 4}	(1.28976, -0.523599)	-2.24593	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1723	9	2/3	1.875	{2, 7, 5, 6}	(1.10715, -2.0944)	2.63623	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1724	9	2/3	1.875	{7, 2, 6, 5}	(1.10715, -1.0472)	-2.63623	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1725	9	2/3	1.875	{2, 3, 6, 5}	(1.5708, -2.0944)	2.0944	000010111 000001111 000001001 000000011 100000111 011000000 110010000 110110000 111110000	{8}
1726	9	2/3	1.875	{6, 4, 0, 7}	(1.28976, 0.523599)	-2.24593	000010111 000001001 000000111 000000011 100000110 010000001 101010001 101110001 111101110	{4}
1727	9	2/3	1.875	{8, 5, 6, 7}	(1.10715, 0)	-2.63623	000010011 000001101 000001100 000000111 100000011 011000000 011100000 100110000 110110000	{4}
1728	9	2/3	1.875	{4, 7, 6, 0}	(1.0472, -0.523599)	-3.14159	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1729	9	2/3	1.875	{0, 6, 7, 4}	(1.0472, 2.61799)	3.14159	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1730	9	2/3	1.875	{5, 6, 2, 7}	(1.10715, 2.0944)	2.63623	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1731	9	2/3	1.875	{6, 5, 7, 2}	(1.10715, 1.0472)	-2.63623	000010001 000001111 000001011 000000111 100000001 011000001 010100000 011100001 111111010	{4}
1732	9	2/3	1.875	{6, 0, 4, 7}	(1.0472, 0.523599)	-3.14159	000010111 000001011 000000111 000000110 100000001 010000011 101100001 111101001 111011110	{4}
1733	9	2/3	1.83333	{0, 3, 1, 2}	(0, 0)	2.0944	001011111 000111111 100011100 010000011 111001111 111010111 111011011 110111101 110111110	{4}
1734	9	2/3	1.83333	{1, 2, 1, 2}	(0, 0)	0	000001111 000000101 000000101 000000010 000000001 100000010 111000011 100101100 111010100	{32}
1735	9	2/3	1.83333	{3, 0, 2, 1}	(0, 0)	-2.0944	001011111 000111111 100011100 010000011 111001111 111010111 111011011 110111101 110111110	{4}
1736	9	2/3	1.83333	{3, 0, 1, 2}	(1.5708, 1.0472)	-3.14159	001011111 000111111 100011100 010000011 111001111 111010111 111011011 110111101 110111110	{4}
1737	9	2/3	1.83333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001111 000000101 000000101 000000010 000000001 100000010 111000011 100101100 111010100	{32}
1738	9	2/3	1.83333	{0, 3, 2, 1}	(1.5708, -1.0472)	3.14159	001011111 000111111 100011100 010000011 111001111 111010111 111011011 110111101 110111110	{4}
1739	9	2/3	1.77778	{3, 4, 3, 4}	(0, 0)	0	000001101 000000111 000000101 000000011 000000011 100000010 111000000 010111000 111110000	{60}
1740	9	2/3	1.77778	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001101 000000111 000000101 000000011 000000011 100000010 111000000 010111000 111110000	{60}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1741	9	2/3	1.75	{1, 4, 2, 4}	(0, 0)	2.0944	000010101 000001011 000001011 000000110 100000100 011000011 100110001 011101001 111001110	{86}
1742	9	2/3	1.75	{4, 1, 4, 2}	(0, 0)	-2.0944	000010101 000001011 000001011 000000110 100000100 011000011 100110001 011101001 111001110	{86}
1743	9	2/3	1.75	{5, 0, 8, 4}	(0.857072, -1.5708)	-1.44547	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1744	9	2/3	1.75	{0, 5, 4, 8}	(0.857072, -1.5708)	1.44547	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1745	9	2/3	1.75	{4, 8, 0, 5}	(0.857072, 1.5708)	1.44547	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1746	9	2/3	1.75	{3, 8, 4, 7}	(0.588003, 3.14159)	-2.24593	000011101 000011011 000010111 000001100 111000011 110100011 101100011 011011100 111011100	{4}
1747	9	2/3	1.75	{8, 4, 5, 0}	(0.857072, 1.5708)	-1.44547	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1748	9	2/3	1.75	{8, 3, 7, 4}	(0.588003, 0)	2.24593	000011101 000011011 000010111 000001100 111000011 110100011 101100011 011011100 111011100	{4}
1749	9	2/3	1.75	{8, 3, 4, 7}	(1.5708, 2.0944)	2.0944	000011101 000011011 000010111 000001100 111000011 110100011 101100011 011011100 111011100	{4}
1750	9	2/3	1.75	{8, 4, 0, 5}	(1.0472, -2.61799)	3.14159	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1751	9	2/3	1.75	{3, 8, 7, 4}	(1.5708, -2.0944)	2.0944	000011101 000011011 000010111 000001100 111000011 110100011 101100011 011011100 111011100	{4}
1752	9	2/3	1.75	{4, 8, 5, 0}	(1.0472, -0.523599)	-3.14159	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1753	9	2/3	1.75	{0, 5, 8, 4}	(1.0472, 2.61799)	3.14159	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1754	9	2/3	1.75	{5, 0, 4, 8}	(1.0472, 0.523599)	-3.14159	000011111 000001101 000001011 000000111 100000110 111000110 110111001 101111001 111100110	{1}
1755	9	2/3	1.75	{4, 1, 2, 4}	(1.5708, 1.0472)	-3.14159	000010101 000001011 000001011 000000110 100000100 011000011 100110001 011101001 111001110	{86}
1756	9	2/3	1.75	{1, 4, 4, 2}	(1.5708, -1.0472)	3.14159	000010101 000001011 000001011 000000110 100000100 011000011 100110001 011101001 111001110	{86}
1757	9	2/3	1.71429	{1, 2, 1, 2}	(0, 0)	0	000010101 000001010 000001010 000000111 100000101 011000011 100110001 011101001 100111110	{116}
1758	9	2/3	1.71429	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001010 000001010 000000111 100000101 011000011 100110001 011101001 100111110	{116}
1759	9	2/3	1.66667	{2, 3, 2, 3}	(0, 0)	0	000010101 000001111 000000111 000000111 100000010 010000000 111100001 011110001 111100110	{20}
1760	9	2/3	1.66667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001111 000000111 000000111 100000010 010000000 111100001 011110001 111100110	{20}
1761	9	2/3	1.6	{6, 7, 6, 7}	(0, 0)	0	000001111 000001111 000000111 000000111 000000001 110000000 111100000 111100000 111110000	{16}
1762	9	2/3	1.6	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000001111 000001111 000000111 000000111 000000001 110000000 111100000 111100000 111110000	{16}
1763	9	2/3	1.58333	{7, 5, 6, 8}	(1.5708, -2.0944)	-1.0472	000110111 000001111 000001011 100010111 100100111 011000110 110111001 111111001 111110110	{4}
1764	9	2/3	1.58333	{5, 7, 8, 6}	(1.5708, 2.0944)	-1.0472	000110111 000001111 000001011 100010111 100100111 011000110 110111001 111111001 111110110	{4}
1765	9	2/3	1.58333	{5, 7, 6, 8}	(1.10715, 3.14159)	2.63623	000110111 000001111 000001011 100010111 100100111 011000110 110111001 111111001 111110110	{4}
1766	9	2/3	1.58333	{7, 5, 8, 6}	(1.10715, 0)	-2.63623	000110111 000001111 000001011 100010111 100100111 011000110 110111001 111111001 111110110	{4}
1767	9	2/3	1.5	{7, 5, 6, 8}	(1.5708, -2.0944)	-1.0472	000101011 000010111 000001111 100001011 010000111 101100110 011011001 111111001 111110110	{4}
1768	9	2/3	1.5	{5, 7, 8, 6}	(1.5708, 2.0944)	-1.0472	000101011 000010111 000001111 100001011 010000111 101100110 011011001 111111001 111110110	{4}
1769	9	2/3	1.5	{0, 1, 5, 4}	(1.5708, -3.14159)	1.42745	000110101 000101101 000011111 110000111 101000010 011000010 111100001 001111000 111100100	{4}
1770	9	2/3	1.5	{0, 1, 4, 5}	(1.5708, 3.14159)	-1.71414	000110101 000101101 000011111 110000111 101000010 011000010 111100001 001111000 111100100	{4}
1771	9	2/3	1.5	{5, 7, 6, 8}	(1.10715, 3.14159)	2.63623	000101011 000010111 000001111 100001011 010000111 101100110 011011001 111111001 111110110	{4}
1772	9	2/3	1.5	{7, 5, 8, 6}	(1.10715, 0)	-2.63623	000101011 000010111 000001111 100001011 010000111 101100110 011011001 111111001 111110110	{4}
1773	9	2/3	1.42857	{1, 2, 1, 2}	(0, 0)	0	000010101 000001010 000001010 000000110 100000101 011000011 100110001 011101001 100011110	{44}
1774	9	2/3	1.42857	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001010 000001010 000000110 100000101 011000011 100110001 011101001 100011110	{44}
1775	9	2/3	1.375	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000111 000000011 000000011 100000001 011000001 011110001 111111110	{140}
1776	9	2/3	1.375	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000111 000000011 000000011 100000001 011000001 011110001 111111110	{140}
1777	9	2/3	1.27273	{2, 3, 2, 3}	(0, 0)	0	000001011 000000101 000000011 000000011 000000001 100000000 010000001 101100000 111110100	{16}
1778	9	2/3	1.27273	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000000101 000000011 000000011 000000001 100000000 010000001 101100000 111110100	{16}
1779	9	2/3	1.25	{1, 3, 2, 3}	(0, 0)	2.0944	000100001 000011111 000011111 100000000 011001111 011010111 011011011 011011101 111011110	{8}
1780	9	2/3	1.25	{3, 1, 3, 2}	(0, 0)	-2.0944	000100001 000011111 000011111 100000000 011001111 011010111 011011011 011011101 111011110	{8}
1781	9	2/3	1.25	{3, 1, 2, 3}	(1.5708, 1.0472)	-3.14159	000100001 000011111 000011111 100000000 011001111 011010111 011011011 011011101 111011110	{8}
1782	9	2/3	1.25	{1, 3, 3, 2}	(1.5708, -1.0472)	3.14159	000100001 000011111 000011111 100000000 011001111 011010111 011011011 011011101 111011110	{8}
1783	9	2/3	1.2	{1, 2, 1, 2}	(0, 0)	0	000110111 000001000 000001000 100010111 100100111 011000001 100110011 100110101 100111110	{16}
1784	9	2/3	1.2	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000110111 000001000 000001000 100010111 100100111 011000001 100110011 100110101 100111110	{16}
1785	9	2/3	1.11111	{0, 1, 0, 1}	(0, 0)	0	000000111 000000111 000000101 000000011 000000011 000000011 111000010 110111101 111111010	{32}
1786	9	2/3	1.11111	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000111 000000111 000000101 000000011 000000011 000000011 111000010 110111101 111111010	{32}
1787	9	2/3	1.1	{2, 3, 2, 3}	(0, 0)	0	000001111 000000111 000000011 000000011 000000001 100000100 110001001 111100001 111110110	{8}
1788	9	2/3	1.1	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000000111 000000011 000000011 000000001 100000100 110001001 111100001 111110110	{8}
1789	9	2/3	1.09091	{1, 2, 1, 2}	(0, 0)	0	000001010 000000101 000000101 000000001 000000001 100000010 011000010 100001101 011110010	{8}
1790	9	2/3	1.09091	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001010 000000101 000000101 000000001 000000001 100000010 011000010 100001101 011110010	{8}
1791	9	2/3	0.875	{3, 4, 3, 4}	(0, 0)	0	000001001 000000110 000000101 000000010 000000010 100000001 011000000 010110001 101001010	{112}
1792	9	2/3	0.875	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000110 000000101 000000010 000000010 100000001 011000000 010110001 101001010	{112}
1793	9	2/3	0.8	{2, 3, 2, 3}	(0, 0)	0	000010111 000001001 000000001 000000001 100000110 010000000 100010010 100010100 111100000	{8}
1794	9	2/3	0.8	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001001 000000001 000000001 100000110 010000000 100010010 100010100 111100000	{8}
1795	9	2/3	0.75	{3, 4, 3, 4}	(0, 0)	0	000001100 000001010 000000111 000000001 000000001 110000001 101000000 011000000 001111000	{28}
1796	9	2/3	0.75	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001100 000001010 000000111 000000001 000000001 110000001 101000000 011000000 001111000	{28}
1797	9	2/3	0.666667	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000010 001100100 111011000	{28}
1798	9	2/3	0.666667	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000010 001100100 111011000	{28}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1799	9	2/3	0.375	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000010 001100100 111111000	{4}
1800	9	2/3	0.375	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000010 001100100 111111000	{4}
1801	9	3/5	87.4853	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000000111 110001011 111010000 111100011 111110101 111110110	{4}
1802	9	3/5	87.4853	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000000111 110001011 111010000 111100011 111110101 111110110	{4}
1803	9	3/5	8.8541	{0, 5, 6, 1}	(0, 0)	0	000011010 000010110 000001111 000000011 110000011 101000001 011000001 111110001 001111110	{8}
1804	9	3/5	8.8541	{0, 5, 1, 6}	(1.5708, 0)	3.14159	000011010 000010110 000001111 000000011 110000011 101000001 011000001 111110001 001111110	{8}
1805	9	3/5	8.63932	{0, 4, 3, 1}	(0, 0)	0	000101111 000011111 000000011 100001111 010001111 110110100 110111000 111110001 111110010	{8}
1806	9	3/5	8.63932	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	000101111 000011111 000000011 100001111 010001111 110110100 110111000 111110001 111110010	{8}
1807	9	3/5	8.2918	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000101 000000010 000000010 110000000 111000001 110110000 111000100	{8}
1808	9	3/5	8.2918	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000101 000000010 000000010 110000000 111000001 110110000 111000100	{8}
1809	9	3/5	7.54509	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000001 010000001 001000001 100100001 111111110	{8}
1810	9	3/5	7.54509	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000001 010000001 001000001 100100001 111111110	{8}
1811	9	3/5	698.847	{2, 3, 2, 3}	(0, 0)	0	000011011 000010100 000001111 000001111 110000011 101100011 011100011 101111101 101111110	{4}
1812	9	3/5	698.847	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000010100 000001111 000001111 110000011 101100011 011100011 101111101 101111110	{4}
1813	9	3/5	69.541	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100001 111101110	{8}
1814	9	3/5	69.541	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100001 111101110	{8}
1815	9	3/5	6.98684	{6, 7, 6, 7}	(0, 0)	0	000100111 000011111 000011111 100000001 011000110 011000110 111011001 111011001 111100110	{4}
1816	9	3/5	6.98684	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000100111 000011111 000011111 100000001 011000110 011000110 111011001 111011001 111100110	{4}
1817	9	3/5	6.52786	{2, 8, 7, 3}	(0, 0)	0	000010111 000001011 000000010 000000001 100000111 010000011 100010000 111011000 110111000	{8}
1818	9	3/5	6.52786	{2, 8, 3, 7}	(1.5708, 0)	-3.14159	000010111 000001011 000000010 000000001 100000111 010000011 100010000 111011000 110111000	{8}
1819	9	3/5	6.43206	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000001101 110000010 111100011 111100011 111011100 111101100	{4}
1820	9	3/5	6.43206	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001101 110000010 111100011 111100011 111011100 111101100	{4}
1821	9	3/5	6.23607	{1, 6, 5, 3}	(0, 0)	0	000001011 000001000 000000111 000000100 000000011 110000110 001101001 101011001 101010110	{16}
1822	9	3/5	6.23607	{1, 6, 3, 5}	(1.5708, 0)	-3.14159	000001011 000001000 000000111 000000100 000000011 110000110 001101001 101011001 101010110	{16}
1823	9	3/5	6	{0, 4, 5, 1}	(0, 0)	0	000010010 000001010 000000110 000000101 100000001 010000001 001100000 111000000 000111000	{8}
1824	9	3/5	6	{0, 4, 1, 5}	(1.5708, 0)	3.14159	000010010 000001010 000000110 000000101 100000001 010000001 001100000 111000000 000111000	{8}
1825	9	3/5	57.8885	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000101 000000100 110000011 110000011 111100001 110011000 111011100	{4}
1826	9	3/5	57.8885	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000101 000000100 110000011 110000011 111100001 110011000 111011100	{4}
1827	9	3/5	56.8328	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001011 000000100 000000011 111000000 110100000 111010001 111010010	{4}
1828	9	3/5	56.8328	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001011 000000100 000000011 111000000 110100000 111010001 111010010	{4}
1829	9	3/5	52.3607	{0, 1, 0, 1}	(0, 0)	0	000011011 000011011 000000100 000000100 110000010 110000001 001100011 110010100 110001100	{8}
1830	9	3/5	52.3607	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011011 000011011 000000100 000000100 110000010 110000001 001100011 110010100 110001100	{8}
1831	9	3/5	519.385	{2, 3, 2, 3}	(0, 0)	0	000011111 000010011 000001111 000001111 110001111 101110110 101111001 111111001 111110110	{4}
1832	9	3/5	519.385	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000010011 000001111 000001111 110001111 101110110 101111001 111111001 111110110	{4}
1833	9	3/5	519.361	{6, 7, 6, 7}	(0, 0)	0	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{4}
1834	9	3/5	519.361	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{4}
1835	9	3/5	50.5967	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100001 111100010	{8}
1836	9	3/5	50.5967	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100001 111100010	{8}
1837	9	3/5	5.73607	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100001 111110010	{8}
1838	9	3/5	5.73607	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100001 111110010	{8}
1839	9	3/5	48.3607	{2, 6, 7, 3}	(0, 0)	0	000011111 000001110 000000101 000000011 100000001 110000111 111001000 110101000 101111000	{8}
1840	9	3/5	48.3607	{2, 6, 3, 7}	(1.5708, 0)	3.14159	000011111 000001110 000000101 000000011 100000001 110000111 111001000 110101000 101111000	{8}
1841	9	3/5	45.1246	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{8}
1842	9	3/5	45.1246	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{8}
1843	9	3/5	4.87539	{2, 3, 2, 3}	(0, 0)	0	000010100 000001011 000000110 000000110 100000001 010000011 101100001 011101000 010011100	{16}
1844	9	3/5	4.87539	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010100 000001011 000000110 000000110 100000001 010000011 101100001 011101000 010011100	{16}
1845	9	3/5	4.81966	{1, 6, 5, 2}	(0, 0)	0	000010010 000001001 000000101 000000011 100000000 010000001 001000001 100100000 011101100	{16}
1846	9	3/5	4.81966	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010010 000001001 000000101 000000011 100000000 010000001 001000001 100100000 011101100	{16}
1847	9	3/5	4.57295	{2, 3, 2, 3}	(0, 0)	0	000001010 000001001 000000111 000000111 000000100 110000000 001110000 101100001 011100010	{8}
1848	9	3/5	4.57295	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001010 000001001 000000111 000000111 000000100 110000000 001110000 101100001 011100010	{8}
1849	9	3/5	4.42229	{7, 8, 7, 8}	(0, 0)	0	000100111 000011111 000011111 100000000 011001011 011010000 111000011 111010100 111010100	{4}
1850	9	3/5	4.42229	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000100111 000011111 000011111 100000000 011001011 011010000 111000011 111010100 111010100	{4}
1851	9	3/5	4.40149	{0, 1, 4, 3}	(1.5708, 0)	0.96804	000001100 000001010 000000110 000000101 000000011 110000001 101100000 011010000 000111000	{4}
1852	9	3/5	4.40149	{0, 1, 3, 4}	(1.5708, 0)	-2.17355	000001100 000001010 000000110 000000101 000000011 110000001 101100000 011010000 000111000	{4}
1853	9	3/5	4.32714	{2, 3, 2, 3}	(0, 0)	0	000011011 000010100 000001111 000001111 110000110 101100111 011111001 101111001 101101110	{4}
1854	9	3/5	4.32714	{2, 3, 3, 2}	(1.5708, 0)	3.14159	000011011 000010100 000001111 000001111 110000110 101100111 011111001 101111001 101101110	{4}
1855	9	3/5	4.22291	{7, 8, 7, 8}	(0, 0)	0	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{16}
1856	9	3/5	4.22291	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{16}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1857	9	3/5	4.06484	{0, 1, 0, 1}	(0, 0)	0	000111111 000111111 000010111 110001111 111000111 110100010 111110001 111111000 111110100	{4}
1858	9	3/5	4.06484	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000111111 000111111 000010111 110001111 111000111 110100010 111110001 111111000 111110100	{4}
1859	9	3/5	4	{1, 2, 1, 2}	(0, 0)	0	000010001 000001111 000001111 000000011 100000001 011000101 011001001 011100001 111111110	{12}
1860	9	3/5	4	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010001 000001111 000001111 000000011 100000001 011000101 011001001 011100001 111111110	{12}
1861	9	3/5	37.1803	{0, 5, 4, 1}	(0, 0)	0	000010011 000001011 000000101 000000101 100000011 010000011 001100000 110011001 111111010	{8}
1862	9	3/5	37.1803	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	000010011 000001011 000000101 000000101 100000011 010000011 001100000 110011001 111111010	{8}
1863	9	3/5	36.5836	{6, 7, 6, 7}	(0, 0)	0	000101111 000011111 000010001 100001111 011000111 110100110 110111001 110111001 111110110	{4}
1864	9	3/5	36.5836	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000101111 000011111 000010001 100001111 011000111 110100110 110111001 110111001 111110110	{4}
1865	9	3/5	358.885	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000000011 100000111 011000111 111011000 111111001 111111010	{4}
1866	9	3/5	358.885	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000000011 100000111 011000111 111011000 111111001 111111010	{4}
1867	9	3/5	33.4164	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000000110 110000001 111000001 111100011 111100100 111011100	{16}
1868	9	3/5	33.4164	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000000110 110000001 111000001 111100011 111100100 111011100	{16}
1869	9	3/5	33.0344	{0, 4, 5, 1}	(0, 0)	0	000010100 000001100 000000101 000000011 100000011 010000011 111000001 000111000 001111100	{16}
1870	9	3/5	33.0344	{0, 4, 1, 5}	(1.5708, 0)	3.14159	000010100 000001100 000000101 000000011 100000011 010000011 111000001 000111000 001111100	{16}
1871	9	3/5	31.3435	{2, 3, 2, 3}	(0, 0)	0	000001100 000001001 000000111 000000111 000000010 110000111 101101001 001111000 011101100	{8}
1872	9	3/5	31.3435	{2, 3, 3, 2}	(1.5708, 0)	3.14159	000001100 000001001 000000111 000000111 000000010 110000111 101101001 001111000 011101100	{8}
1873	9	3/5	3.92047	{6, 7, 6, 7}	(0, 0)	0	001011110 000110111 100011000 010001111 111000111 101100111 110111001 110111001 010111110	{4}
1874	9	3/5	3.92047	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	001011110 000110111 100011000 010001111 111000111 101100111 110111001 110111001 010111110	{4}
1875	9	3/5	3.76393	{2, 8, 6, 4}	(0, 0)	0	000001100 000001010 000000100 000000011 000000001 110000000 101000001 010100000 000110100	{16}
1876	9	3/5	3.76393	{2, 8, 4, 6}	(1.5708, 0)	-3.14159	000001100 000001010 000000100 000000011 000000001 110000000 101000001 010100000 000110100	{16}
1877	9	3/5	3.5	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000010 100000000 010000001 001000001 100100001 111001110	{8}
1878	9	3/5	3.5	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000010 100000000 010000001 001000001 100100001 111001110	{8}
1879	9	3/5	3.16718	{7, 8, 7, 8}	(0, 0)	0	000101011 000010111 000001111 100000000 010000111 101000000 011010000 111010000 111010000	{4}
1880	9	3/5	3.16718	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000101011 000010111 000001111 100000000 010000111 101000000 011010000 111010000 111010000	{4}
1881	9	3/5	28.9443	{7, 8, 7, 8}	(0, 0)	0	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{12}
1882	9	3/5	28.9443	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{12}
1883	9	3/5	27.1803	{1, 6, 5, 2}	(0, 0)	0	000001101 000001000 000000100 000000011 000000011 110000001 101000001 000110000 100111100	{32}
1884	9	3/5	27.1803	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000001101 000001000 000000100 000000011 000000011 110000001 101000001 000110000 100111100	{32}
1885	9	3/5	26.8713	{6, 7, 6, 7}	(0, 0)	0	000110111 000101001 000011111 110001111 101000110 011100111 101111001 101111001 111101110	{4}
1886	9	3/5	26.8713	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000110111 000101001 000011111 110001111 101000110 011100111 101111001 101111001 111101110	{4}
1887	9	3/5	22.1115	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000011011 000000101 111000011 111000011 110100001 111011000 111111100	{4}
1888	9	3/5	22.1115	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000011011 000000101 111000011 111000011 110100001 111011000 111111100	{4}
1889	9	3/5	20	{1, 2, 1, 2}	(0, 0)	0	000010001 000001111 000001111 000000110 100000001 011000001 011100011 011100101 111011110	{4}
1890	9	3/5	20	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010001 000001111 000001111 000000110 100000001 011000001 011100011 011100101 111011110	{4}
1891	9	3/5	2.99303	{2, 3, 2, 3}	(0, 0)	0	000001010 000001001 000000111 000000111 000000011 110000111 001101000 101111001 011111010	{8}
1892	9	3/5	2.99303	{2, 3, 3, 2}	(1.5708, 0)	3.14159	000001010 000001001 000000111 000000111 000000011 110000111 001101000 101111001 011111010	{8}
1893	9	3/5	2.80902	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000010 100000000 010000000 001000000 100100001 111000010	{8}
1894	9	3/5	2.80902	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000010 100000000 010000000 001000000 100100001 111000010	{8}
1895	9	3/5	2.44721	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100000 111110000	{8}
1896	9	3/5	2.44721	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100000 111110000	{8}
1897	9	3/5	2.22222	{1, 2, 1, 2}	(0, 0)	0	000011110 000001111 000001111 000000111 100001110 111010101 111011001 111110001 011101110	{4}
1898	9	3/5	2.22222	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011110 000001111 000001111 000000011 100001110 111010101 111011001 111110001 011101110	{4}
1899	9	3/5	198.387	{2, 3, 2, 3}	(0, 0)	0	000011110 000010011 000001111 000001111 110000001 101100011 101100001 111101001 011111110	{8}
1900	9	3/5	198.387	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000010011 000001111 000001111 110000001 101100011 101100001 111101001 011111110	{8}
1901	9	3/5	19.9443	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{16}
1902	9	3/5	19.9443	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{16}
1903	9	3/5	18.9443	{2, 3, 2, 3}	(0, 0)	0	000011110 000001000 000000111 000000111 100000111 110000001 101110001 101110001 001111110	{4}
1904	9	3/5	18.9443	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000001000 000000111 000000111 100000111 110000001 101110001 101110001 001111110	{4}
1905	9	3/5	148.79	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000111 000000111 000000011 110000000 111100000 111110001 111110010	{4}
1906	9	3/5	148.79	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000111 000000111 000000011 110000000 111100000 111110001 111110010	{4}
1907	9	3/5	14.0902	{1, 6, 7, 2}	(0, 0)	0	000010111 000001101 000001011 000000110 100000001 011000000 110100000 101100000 111010000	{8}
1908	9	3/5	14.0902	{1, 6, 2, 7}	(1.5708, 0)	3.14159	000010111 000001101 000001011 000000110 100000001 011000000 110100000 101100000 111010000	{8}
1909	9	3/5	125.374	{7, 8, 7, 8}	(0, 0)	0	000101111 000011111 000011111 100000011 011001100 111010011 111010011 111101100 111101100	{4}
1910	9	3/5	125.374	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000101111 000011111 000011111 100000011 011001100 111010011 111010011 111101100 111101100	{4}
1911	9	3/5	118.138	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
1912	9	3/5	118.138	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
1913	9	3/5	11.9721	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000110 000000101 000000011 110000000 001100000 111010001 110110010	{8}
1914	9	3/5	11.9721	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000110 000000101 000000011 110000000 001100000 111010001 110110010	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1915	9	3/5	11.0557	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000101 100000010 111000011 111100001 111011000 111101100	{8}
1916	9	3/5	11.0557	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000101 100000010 111000011 111100001 111011000 111101100	{8}
1917	9	3/5	10.4721	{2, 3, 2, 3}	(0, 0)	0	000011010 000010101 000001111 000001111 110000011 101100010 011100001 101111000 011110100	{20}
1918	9	3/5	10.4721	{2, 3, 3, 2}	(1.5708, 0)	3.14159	000011010 000010101 000001111 000001111 110000011 101100010 011100001 101111000 011110100	{20}
1919	9	3/5	1.95492	{1, 6, 5, 2}	(0, 0)	0	000010010 000001001 000000101 000000011 100000001 010000001 001000001 100100000 011111100	{8}
1920	9	3/5	1.95492	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010010 000001001 000000101 000000011 100000001 010000001 001000001 100100000 011111100	{8}
1921	9	3/5	1.94857	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000011 100000011 111000110 111001001 111111001 111110110	{12}
1922	9	3/5	1.94857	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000011 100000011 111000110 111001001 111111001 111110110	{12}
1923	9	3/5	1.86223	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000010111 000001010 111000101 110100111 111011001 111101000 111011100	{4}
1924	9	3/5	1.86223	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000010111 000001010 111000101 110100111 111011001 111101000 111011100	{4}
1925	9	3/5	1.7486	{2, 3, 2, 3}	(0, 0)	0	000011111 000011111 000001111 000001111 110000011 111100010 111100001 111111000 111110100	{8}
1926	9	3/5	1.7486	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001111 110000011 111100010 111100001 111111000 111110100	{8}
1927	9	3/5	1.69098	{1, 6, 5, 2}	(0, 0)	0	000010010 000001001 000000101 000000011 100000001 010000000 001000000 100100000 011110000	{8}
1928	9	3/5	1.69098	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010010 000001001 000000101 000000011 100000001 010000000 001000000 100100000 011110000	{8}
1929	9	3/5	1.65836	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001010 000000111 000000111 111000000 110110001 111110000 110110100	{4}
1930	9	3/5	1.65836	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001010 000000111 000000111 111000000 110110001 111110000 110110100	{4}
1931	9	1/2	9.66667	{0, 1, 0, 1}	(0, 0)	0	000010011 000010011 000001111 000001101 110000100 001100010 001110001 111001001 111100110	{8}
1932	9	1/2	9.66667	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010011 000010011 000001111 000001101 110000100 001100010 001110001 111001001 111100110	{8}
1933	9	1/2	9.5	{1, 2, 1, 2}	(0, 0)	0	000001110 000001101 000001101 000000011 000000011 111000011 111000010 100111101 011111010	{112}
1934	9	1/2	9.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001110 000001101 000001101 000000011 000000011 111000011 111000010 100111101 011111010	{112}
1935	9	1/2	9	{6, 0, 2, 1}	(0, 0)	1.5708	000001110 000001101 000001110 000000011 000000001 110000010 111000001 101101001 010110110	{200}
1936	9	1/2	9	{1, 2, 1, 2}	(0, 0)	0	000001110 000000101 000000101 000000011 000000011 100000011 111000010 100111101 011111010	{100}
1937	9	1/2	9	{0, 6, 1, 2}	(0, 0)	-1.5708	000001110 000001101 000000110 000000011 000000001 110000010 111000001 101101001 010110110	{200}
1938	9	1/2	9	{0, 6, 2, 1}	(1.5708, 0.785398)	-3.14159	000001110 000001101 000000110 000000011 000000001 110000010 111000001 101101001 010110110	{200}
1939	9	1/2	9	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001110 000000101 000000101 000000011 000000011 100000011 111000010 100111101 011111010	{100}
1940	9	1/2	9	{6, 0, 1, 2}	(1.5708, -0.785398)	3.14159	000001110 000001101 000000110 000000011 000000001 110000010 111000001 101101001 010110110	{200}
1941	9	1/2	8	{7, 0, 3, 2}	(0, 0)	1.5708	000001110 000001001 000000111 000000011 000000001 110000010 101000001 101101000 011110100	{360}
1942	9	1/2	8	{0, 1, 0, 1}	(0, 0)	0	000001001 000001001 000000111 000000110 000000110 110000010 001110001 001111000 111000100	{136}
1943	9	1/2	8	{5, 2, 3, 8}	(0, 0)	3.14159	000110111 000011111 000001010 100000011 110000111 011000101 110011000 111110001 110111010	{3}
1944	9	1/2	8	{0, 7, 2, 3}	(0, 0)	-1.5708	000001110 000001001 000000111 000000011 000000001 110000010 101000001 101101000 011110100	{360}
1945	9	1/2	8	{0, 7, 3, 2}	(1.5708, 0.785398)	-3.14159	000001110 000001001 000000111 000000011 000000001 110000010 101000001 101101000 011110100	{360}
1946	9	1/2	8	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001001 000001001 000000111 000000110 000000110 110000010 001110001 001111000 111000100	{136}
1947	9	1/2	8	{2, 5, 3, 8}	(1.5708, 1.5708)	-3.14159	000110111 000011111 000001010 100000011 110000111 011000101 110011000 111110001 110111010	{3}
1948	9	1/2	8	{7, 0, 2, 3}	(1.5708, -0.785398)	3.14159	000001110 000001001 000000111 000000011 000000001 110000010 101000001 101101000 011110100	{360}
1949	9	1/2	7	{6, 0, 2, 3}	(0, 0)	1.5708	000001100 000001010 000000111 000000101 000000011 110000001 101100011 011010101 001111110	{4968}
1950	9	1/2	7	{0, 6, 3, 2}	(0, 0)	-1.5708	000001100 000001010 000000111 000000101 000000011 110000001 101100011 011010101 001111110	{4968}
1951	9	1/2	7	{0, 6, 2, 3}	(1.5708, 0.785398)	-3.14159	000001100 000001010 000000111 000000101 000000011 110000001 101100011 011010101 001111110	{4968}
1952	9	1/2	7	{6, 0, 3, 2}	(1.5708, -0.785398)	3.14159	000001100 000001010 000000111 000000101 000000011 110000001 101100011 011010101 001111110	{4968}
1953	9	1/2	6.5	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000111 000000011 110000001 001100001 111110000 111111100	{602}
1954	9	1/2	6.5	{3, 6, 4, 6}	(0, 0)	-3.14159	000010101 000001110 000001110 000000011 100000011 011000001 111000001 011110001 100111110	{65}
1955	9	1/2	6.5	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000111 000000011 110000001 001100001 111110000 111111100	{602}
1956	9	1/2	6.5	{6, 3, 4, 6}	(1.5708, -1.5708)	3.14159	000010101 000001110 000001110 000000011 100000011 011000001 111000001 011110001 100111110	{65}
1957	9	1/2	6	{2, 0, 2, 5}	(0, 0)	1.5708	000001100 000000110 000000101 000000011 000000001 100000010 111000001 010101001 001110110	{6122}
1958	9	1/2	6	{0, 2, 5, 2}	(0, 0)	-1.5708	000001100 000000110 000000101 000000011 000000001 100000010 111000001 010101001 001110110	{6122}
1959	9	1/2	6	{0, 2, 2, 5}	(1.5708, 0.785398)	-3.14159	000001100 000000110 000000101 000000011 000000001 100000010 111000001 010101001 001110110	{6122}
1960	9	1/2	6	{2, 0, 5, 2}	(1.5708, -0.785398)	3.14159	000001100 000000110 000000101 000000011 000000001 100000010 111000001 010101001 001110110	{6122}
1961	9	1/2	5.66667	{2, 3, 2, 3}	(0, 0)	0	000001100 000001011 000000111 000000111 000000011 110000001 101100001 011110000 011111100	{48}
1962	9	1/2	5.66667	{6, 1, 5, 1}	(0, 0)	-3.14159	000011100 000010011 000001111 000000111 000000011 110000101 101000001 011100000 011111100	{10}
1963	9	1/2	5.66667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000111 000000011 110000001 101100001 011110000 011111100	{48}
1964	9	1/2	5.66667	{1, 6, 5, 1}	(1.5708, -1.5708)	3.14159	000011100 000010011 000001111 000000011 110000101 101000001 101010001 011100000 011111100	{10}
1965	9	1/2	5.5	{1, 2, 1, 2}	(0, 0)	0	000000110 000000101 000000101 000000011 000000011 000000001 111000000 100110000 011111000	{464}
1966	9	1/2	5.5	{7, 2, 6, 2}	(0, 0)	-3.14159	000001110 000001110 000001001 000000111 000000001 111000010 110100001 110101001 001110110	{44}
1967	9	1/2	5.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000110 000000101 000000101 000000011 000000011 000000001 111000000 100110000 011111000	{464}
1968	9	1/2	5.5	{2, 7, 6, 2}	(1.5708, -1.5708)	3.14159	000001110 000001110 000001001 000000111 000000001 111000010 110100001 110101001 001110110	{44}
1969	9	1/2	4.75	{2, 3, 2, 3}	(0, 0)	0	000011111 000001111 000000011 000000011 100000110 110000001 110010001 111110000 111101100	{154}
1970	9	1/2	4.75	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001111 000000011 000000011 100000110 110000001 110010001 111110000 111101100	{154}
1971	9	1/2	4.66667	{2, 3, 2, 3}	(0, 0)	0	000001111 000001101 000000011 000000011 000000010 110000010 110000001 101111000 111100100	{148}
1972	9	1/2	4.66667	{0, 4, 8, 4}	(0, 0)	-3.14159	000010100 000001111 000001101 000000010 100000011 011000010 111000001 010111001 011010110	{9}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
1973	9	1/2	4.66667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000001101 000000011 000000011 000000010 110000010 110000001 101111000 111100100	{148}
1974	9	1/2	4.66667	{4, 0, 8, 4}	(1.5708, -1.5708)	3.14159	000010100 000001111 000001101 000000010 100000011 011000010 111000001 010111001 011010110	{9}
1975	9	1/2	4.5	{1, 2, 1, 2}	(0, 0)	0	000001011 000000101 000000101 000000011 000000011 100000001 011000010 100110101 111111010	{1602}
1976	9	1/2	4.5	{1, 5, 2, 5}	(0, 0)	-3.14159	000001111 000000101 000000101 000000011 000000011 100000001 111000001 100110000 111111100	{392}
1977	9	1/2	4.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000101 000000101 000000011 000000011 100000001 011000010 100110101 111111010	{1602}
1978	9	1/2	4.5	{5, 1, 2, 5}	(1.5708, -1.5708)	3.14159	000001111 000000101 000000101 000000011 000000011 100000001 111000001 100110000 111111100	{392}
1979	9	1/2	4.4	{5, 2, 0, 3}	(0.61548, 2.35619)	1.77215	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1980	9	1/2	4.4	{0, 3, 5, 2}	(0.61548, -2.35619)	1.77215	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1981	9	1/2	4.4	{3, 0, 2, 5}	(0.61548, -0.785398)	-1.77215	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1982	9	1/2	4.4	{2, 5, 3, 0}	(0.61548, 0.785398)	-1.77215	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1983	9	1/2	4.4	{2, 5, 0, 3}	(1.23096, -2.35619)	-2.49809	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1984	9	1/2	4.4	{5, 2, 3, 0}	(1.23096, -0.785398)	2.49809	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1985	9	1/2	4.4	{0, 3, 2, 5}	(1.23096, 2.35619)	-2.49809	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1986	9	1/2	4.4	{3, 0, 5, 2}	(1.23096, 0.785398)	2.49809	000101110 000011101 000010111 100000011 011001111 110010110 111011001 101111001 011110110	{1}
1987	9	1/2	4.33333	{1, 6, 7, 5}	(0, 0)	1.5708	000001011 000001001 000000111 000000111 000000111 110000110 001111001 101111000 111110100	{16}
1988	9	1/2	4.33333	{0, 5, 4, 1}	(0, 0)	0	000010011 000001011 000000111 000000100 100000011 010000011 001100001 111011000 111011100	{16}
1989	9	1/2	4.33333	{7, 2, 5, 3}	(0, 0)	-3.14159	000001111 000001010 000001001 000000110 000000101 111000001 100110000 110100001 101011010	{4}
1990	9	1/2	4.33333	{6, 1, 5, 7}	(0, 0)	-1.5708	000001011 000001001 000000111 000000111 000000111 110000110 001111001 101111000 111110100	{16}
1991	9	1/2	4.33333	{6, 1, 7, 5}	(1.5708, 0.785398)	-3.14159	000001011 000001001 000000111 000000111 000000111 110000110 001111001 101111000 111110100	{16}
1992	9	1/2	4.33333	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	000010011 000001011 000000111 000000100 100000011 010000011 001100001 111011000 111011100	{16}
1993	9	1/2	4.33333	{2, 7, 5, 3}	(1.5708, -1.5708)	3.14159	000001111 000001010 000001001 000000110 000000101 111000001 100110000 110100001 101011010	{4}
1994	9	1/2	4.33333	{1, 6, 5, 7}	(1.5708, -0.785398)	3.14159	000001011 000001001 000000111 000000111 000000111 110000110 001111001 101111000 111110100	{16}
1995	9	1/2	4.2	{5, 6, 1, 3}	(0.61548, 2.35619)	1.77215	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
1996	9	1/2	4.2	{1, 3, 5, 6}	(0.61548, -2.35619)	1.77215	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
1997	9	1/2	4.2	{3, 1, 6, 5}	(0.61548, -0.785398)	-1.77215	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
1998	9	1/2	4.2	{6, 5, 3, 1}	(0.61548, 0.785398)	-1.77215	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
1999	9	1/2	4.2	{6, 5, 1, 3}	(1.23096, -2.35619)	-2.49809	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
2000	9	1/2	4.2	{5, 6, 3, 1}	(1.23096, -0.785398)	2.49809	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
2001	9	1/2	4.2	{1, 3, 6, 5}	(1.23096, 2.35619)	-2.49809	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
2002	9	1/2	4.2	{3, 1, 5, 6}	(1.23096, 0.785398)	2.49809	001011111 000110110 100011100 010001111 111000111 101100011 111110001 110111001 100111110	{1}
2003	9	1/2	3.8	{2, 3, 2, 3}	(0, 0)	0	000011000 000001101 000000110 000000110 100000011 110000011 011100001 001111001 010011110	{32}
2004	9	1/2	3.8	{1, 5, 0, 3}	(0.61548, 2.35619)	1.77215	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2005	9	1/2	3.8	{0, 6, 1, 2}	(0.61548, -2.35619)	-1.77215	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2006	9	1/2	3.8	{0, 3, 1, 5}	(0.61548, -2.35619)	1.77215	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2007	9	1/2	3.8	{3, 0, 5, 1}	(0.61548, -0.785398)	-1.77215	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2008	9	1/2	3.8	{5, 1, 3, 0}	(0.61548, 0.785398)	-1.77215	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2009	9	1/2	3.8	{6, 0, 2, 1}	(0.61548, -0.785398)	1.77215	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2010	9	1/2	3.8	{1, 2, 0, 6}	(0.61548, 2.35619)	-1.77215	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2011	9	1/2	3.8	{2, 1, 6, 0}	(0.61548, 0.785398)	1.77215	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2012	9	1/2	3.8	{6, 0, 1, 2}	(1.23096, 2.35619)	2.49809	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2013	9	1/2	3.8	{5, 1, 0, 3}	(1.23096, -2.35619)	-2.49809	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2014	9	1/2	3.8	{1, 2, 6, 0}	(1.23096, -2.35619)	2.49809	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2015	9	1/2	3.8	{2, 1, 0, 6}	(1.23096, -0.785398)	-2.49809	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2016	9	1/2	3.8	{0, 6, 2, 1}	(1.23096, 0.785398)	-2.49809	000011111 000010101 000001110 000001011 110000011 101100011 111000000 101111001 110111010	{2}
2017	9	1/2	3.8	{1, 5, 3, 0}	(1.23096, -0.785398)	2.49809	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2018	9	1/2	3.8	{0, 3, 5, 1}	(1.23096, 2.35619)	-2.49809	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2019	9	1/2	3.8	{3, 0, 1, 5}	(1.23096, 0.785398)	2.49809	000110011 000011111 000001101 100000111 110000011 011000000 011100011 110110101 111110110	{1}
2020	9	1/2	3.8	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011000 000001101 000000110 000000110 100000011 011100001 001111001 010011110	{32}
2021	9	1/2	3.75	{8, 3, 6, 5}	(0, 0)	1.5708	000011010 000010101 000001111 000000111 110000110 101000001 011110000 101110001 011101010	{80}
2022	9	1/2	3.75	{2, 3, 2, 3}	(0, 0)	0	000001100 000001011 000000111 000000111 000000011 110000001 101100000 011110000 011111000	{504}
2023	9	1/2	3.75	{1, 5, 2, 5}	(0, 0)	-3.14159	000010101 000001111 000001100 000000011 100000011 011000011 111000000 010111000 011011000	{81}
2024	9	1/2	3.75	{3, 8, 5, 6}	(0, 0)	-1.5708	000011010 000010101 000001111 000000111 110000110 101000001 011110000 101110001 011101010	{80}
2025	9	1/2	3.75	{3, 8, 6, 5}	(1.5708, 0.785398)	-3.14159	000011010 000010101 000001111 000000111 110000110 101000001 011110000 101110001 011101010	{80}
2026	9	1/2	3.75	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000111 000000011 110000001 101100000 011110000 011111000	{504}
2027	9	1/2	3.75	{5, 1, 2, 5}	(1.5708, -1.5708)	3.14159	000010101 000001111 000001100 000000011 100000011 011000011 111000000 010111000 110111000	{81}
2028	9	1/2	3.75	{8, 3, 5, 6}	(1.5708, -0.785398)	3.14159	000011010 000010101 000001111 000000111 110000110 101000001 011110000 101110001 011101010	{80}
2029	9	1/2	3.66667	{4, 8, 6, 5}	(0, 0)	1.5708	000001110 000000111 000000111 000000111 000000011 100000011 111100001 111111000 011111100	{4}
2030	9	1/2	3.66667	{3, 4, 3, 4}	(0, 0)	0	000001101 000001001 000000100 000000011 000000011 110000011 101000010 000111100 110111000	{256}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2031	9	1/2	3.66667	{3, 5, 4, 5}	(0, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000001 101000000 110110001 101111010	{84}
2032	9	1/2	3.66667	{8, 4, 5, 6}	(0, 0)	-1.5708	000001110 000000111 000000111 000000111 000000011 100000011 111100001 111111000 011111100	{4}
2033	9	1/2	3.66667	{8, 4, 6, 5}	(1.5708, 0.785398)	-3.14159	000001110 000000111 000000111 000000111 000000011 100000011 111100001 111111000 011111100	{4}
2034	9	1/2	3.66667	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001101 000001001 000000100 000000011 000000011 110000011 101000010 000111100 110111000	{256}
2035	9	1/2	3.66667	{5, 3, 4, 5}	(1.5708, -1.5708)	3.14159	000001111 000001010 000000101 000000011 000000011 110000001 101000000 110110001 101111010	{84}
2036	9	1/2	3.66667	{4, 8, 5, 6}	(1.5708, -0.785398)	3.14159	000001110 000000111 000000111 000000111 000000011 100000011 111100001 111111000 011111100	{4}
2037	9	1/2	3.6	{4, 3, 1, 5}	(0.61548, 2.35619)	1.77215	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2038	9	1/2	3.6	{1, 5, 4, 3}	(0.61548, -2.35619)	1.77215	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2039	9	1/2	3.6	{5, 1, 3, 4}	(0.61548, -0.785398)	-1.77215	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2040	9	1/2	3.6	{3, 4, 5, 1}	(0.61548, 0.785398)	-1.77215	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2041	9	1/2	3.6	{3, 4, 1, 5}	(1.23096, -2.35619)	-2.49809	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2042	9	1/2	3.6	{4, 3, 5, 1}	(1.23096, -0.785398)	2.49809	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2043	9	1/2	3.6	{1, 5, 3, 4}	(1.23096, 2.35619)	-2.49809	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2044	9	1/2	3.6	{5, 1, 4, 3}	(1.23096, 0.785398)	2.49809	000101101 000011011 000001010 100000111 010000101 111000011 100110001 011101000 110111100	{2}
2045	9	1/2	3.5	{2, 4, 5, 3}	(0, 0)	0	000000111 000000111 000000101 000000100 000000011 000000010 111100000 110011000 111010000	{2912}
2046	9	1/2	3.5	{7, 1, 4, 5}	(0, 0)	-3.14159	000001111 000000100 000000011 000000011 000000010 100000001 110000001 101110000 101101100	{781}
2047	9	1/2	3.5	{2, 4, 3, 5}	(1.5708, 0)	-3.14159	000000111 000000111 000000101 000000100 000000011 000000010 111100000 110011000 111010000	{2912}
2048	9	1/2	3.5	{1, 7, 4, 5}	(1.5708, -1.5708)	3.14159	000001111 000000100 000000011 000000011 000000010 100000001 110000001 101110000 101101100	{781}
2049	9	1/2	3.4	{1, 4, 3, 2}	(0, 0)	0	000101010 000010110 000001111 100000111 010000101 101000001 011110000 111100001 001111010	{16}
2050	9	1/2	3.4	{1, 4, 2, 3}	(1.5708, 0)	-3.14159	000101010 000010110 000001111 100000111 010000101 101000001 011110000 111100001 001111010	{16}
2051	9	1/2	3.33333	{0, 7, 5, 6}	(0, 0)	1.5708	000001110 000000111 000000111 000000111 000000001 100000001 111100001 111100000 011111100	{16}
2052	9	1/2	3.33333	{0, 7, 1, 6}	(0, 0)	0	000001011 000001010 000000111 000000110 000000001 110000001 001100001 111100001 101011110	{296}
2053	9	1/2	3.33333	{7, 2, 3, 5}	(0, 0)	-3.14159	000001111 000001110 000000111 000000011 000000001 110000000 111000001 111100000 101110100	{16}
2054	9	1/2	3.33333	{7, 0, 6, 5}	(0, 0)	-1.5708	000001110 000000111 000000111 000000111 000000001 100000001 111100001 111100000 011111100	{16}
2055	9	1/2	3.33333	{7, 0, 5, 6}	(1.5708, 0.785398)	-3.14159	000001110 000000111 000000111 000000111 000000001 100000001 111100001 111100000 011111100	{16}
2056	9	1/2	3.33333	{0, 7, 6, 1}	(1.5708, 0)	-3.14159	000001011 000001010 000000111 000000110 000000001 110000001 001100001 111100001 101011110	{296}
2057	9	1/2	3.33333	{2, 7, 3, 5}	(1.5708, -1.5708)	3.14159	000001111 000001110 000000111 000000011 000000001 110000000 111000001 111100000 101110100	{16}
2058	9	1/2	3.33333	{0, 7, 6, 5}	(1.5708, -0.785398)	3.14159	000001110 000000111 000000111 000000111 000000001 100000001 111100001 111100000 011111100	{16}
2059	9	1/2	3.2	{2, 3, 2, 3}	(0, 0)	0	000010110 000001010 000000111 000000111 100000011 010000001 101100001 111110001 001111110	{76}
2060	9	1/2	3.2	{1, 4, 2, 3}	(0.61548, 2.35619)	1.77215	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2061	9	1/2	3.2	{5, 6, 1, 2}	(0.61548, -2.35619)	-1.77215	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2062	9	1/2	3.2	{2, 3, 1, 4}	(0.61548, -2.35619)	1.77215	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2063	9	1/2	3.2	{3, 2, 4, 1}	(0.61548, -0.785398)	-1.77215	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2064	9	1/2	3.2	{4, 1, 3, 2}	(0.61548, 0.785398)	-1.77215	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2065	9	1/2	3.2	{6, 5, 2, 1}	(0.61548, -0.785398)	1.77215	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2066	9	1/2	3.2	{1, 2, 5, 6}	(0.61548, 2.35619)	-1.77215	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2067	9	1/2	3.2	{2, 1, 6, 5}	(0.61548, 0.785398)	1.77215	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2068	9	1/2	3.2	{6, 5, 1, 2}	(1.23096, 2.35619)	2.49809	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2069	9	1/2	3.2	{4, 1, 2, 3}	(1.23096, -2.35619)	-2.49809	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2070	9	1/2	3.2	{1, 2, 6, 5}	(1.23096, -2.35619)	2.49809	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2071	9	1/2	3.2	{2, 1, 5, 6}	(1.23096, -0.785398)	-2.49809	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2072	9	1/2	3.2	{5, 6, 2, 1}	(1.23096, 0.785398)	-2.49809	000101111 000011011 000000011 100001111 010000110 110100101 100111001 111110001 111101110	{2}
2073	9	1/2	3.2	{1, 4, 3, 2}	(1.23096, -0.785398)	2.49809	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2074	9	1/2	3.2	{2, 3, 4, 1}	(1.23096, 2.35619)	-2.49809	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2075	9	1/2	3.2	{3, 2, 1, 4}	(1.23096, 0.785398)	2.49809	000101011 000011111 000010101 100000110 011001011 110010111 011101001 110111001 111011110	{1}
2076	9	1/2	3.2	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010110 000001010 000000111 000000111 100000011 010000001 101100001 111110001 001111110	{76}
2077	9	1/2	3	{0, 4, 0, 3}	(0, 0)	0.643501	001011111 000101111 100010111 010001000 101000100 110100011 111010011 111001101 111001110	{6}
2078	9	1/2	3	{4, 0, 3, 0}	(0, 0)	-0.643501	001011111 000101111 100010111 010001000 101000100 110100011 111010011 111001101 111001110	{6}
2079	9	1/2	3	{2, 6, 0, 3}	(0.61548, 2.35619)	1.77215	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2080	9	1/2	3	{0, 6, 1, 2}	(0.61548, -2.35619)	-1.77215	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2081	9	1/2	3	{0, 3, 2, 6}	(0.61548, -2.35619)	1.77215	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2082	9	1/2	3	{3, 0, 6, 2}	(0.61548, -0.785398)	-1.77215	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2083	9	1/2	3	{6, 2, 3, 0}	(0.61548, 0.785398)	-1.77215	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2084	9	1/2	3	{6, 0, 2, 1}	(0.61548, -0.785398)	1.77215	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2085	9	1/2	3	{1, 2, 0, 6}	(0.61548, 2.35619)	-1.77215	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2086	9	1/2	3	{2, 1, 6, 0}	(0.61548, 0.785398)	1.77215	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2087	9	1/2	3	{6, 0, 1, 2}	(1.23096, 2.35619)	2.49809	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2088	9	1/2	3	{6, 2, 0, 3}	(1.23096, -2.35619)	-2.49809	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2089	9	1/2	3	{1, 2, 6, 0}	(1.23096, -2.35619)	2.49809	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2090	9	1/2	3	{2, 1, 0, 6}	(1.23096, -0.785398)	-2.49809	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2091	9	1/2	3	{0, 6, 2, 1}	(1.23096, 0.785398)	-2.49809	000011111 000010111 000001100 000001001 110000011 101100011 111000000 110011001 110111010	{5}
2092	9	1/2	3	{2, 6, 3, 0}	(1.23096, -0.785398)	2.49809	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2093	9	1/2	3	{0, 3, 6, 2}	(1.23096, 2.35619)	-2.49809	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2094	9	1/2	3	{3, 0, 2, 6}	(1.23096, 0.785398)	2.49809	000101001 000010111 000001111 100000011 010000110 101000001 011010000 011110001 111101010	{4}
2095	9	1/2	3	{4, 0, 0, 3}	(1.5708, 0.321751)	-3.14159	001011111 000101111 100010111 010001000 101000100 110100011 111010011 111001101 111001110	{6}
2096	9	1/2	3	{0, 4, 3, 0}	(1.5708, -0.321751)	3.14159	001011111 000101111 100010111 010001000 101000100 110100011 111010011 111001101 111001110	{6}
2097	9	1/2	2.83333	{2, 3, 2, 3}	(0, 0)	0	000001101 000001001 000000111 000000111 000000011 110000011 101100001 001111000 111111100	{64}
2098	9	1/2	2.83333	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001001 000000111 000000111 000000011 110000011 101100001 001111000 111111100	{64}
2099	9	1/2	2.8	{2, 3, 2, 3}	(0, 0)	0	000001110 000001101 000000011 000000011 000000010 110000001 110000000 101110000 011101000	{220}
2100	9	1/2	2.8	{0, 2, 7, 3}	(0.61548, 2.35619)	1.77215	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2101	9	1/2	2.8	{1, 5, 2, 3}	(0.61548, -2.35619)	-1.77215	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2102	9	1/2	2.8	{7, 3, 0, 2}	(0.61548, -2.35619)	1.77215	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2103	9	1/2	2.8	{3, 7, 2, 0}	(0.61548, -0.785398)	-1.77215	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2104	9	1/2	2.8	{2, 0, 3, 7}	(0.61548, 0.785398)	-1.77215	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2105	9	1/2	2.8	{5, 1, 3, 2}	(0.61548, -0.785398)	1.77215	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2106	9	1/2	2.8	{2, 3, 1, 5}	(0.61548, 2.35619)	-1.77215	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2107	9	1/2	2.8	{3, 2, 5, 1}	(0.61548, 0.785398)	1.77215	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2108	9	1/2	2.8	{5, 1, 2, 3}	(1.23096, 2.35619)	2.49809	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2109	9	1/2	2.8	{2, 0, 7, 3}	(1.23096, -2.35619)	-2.49809	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2110	9	1/2	2.8	{2, 3, 5, 1}	(1.23096, -2.35619)	2.49809	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2111	9	1/2	2.8	{3, 2, 1, 5}	(1.23096, -0.785398)	-2.49809	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2112	9	1/2	2.8	{1, 5, 3, 2}	(1.23096, 0.785398)	-2.49809	000010111 000001111 000001110 000001001 100000001 011100000 111000011 111000101 110110110	{5}
2113	9	1/2	2.8	{0, 2, 3, 7}	(1.23096, -0.785398)	2.49809	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2114	9	1/2	2.8	{7, 3, 2, 0}	(1.23096, 2.35619)	-2.49809	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2115	9	1/2	2.8	{3, 7, 0, 2}	(1.23096, 0.785398)	2.49809	000010111 000010100 000001111 000001011 110000110 001100001 111010000 101110001 101101010	{9}
2116	9	1/2	2.8	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001110 000001101 000000011 000000011 000000010 110000001 110000000 101110000 011101000	{220}
2117	9	1/2	2.75	{0, 6, 4, 5}	(0, 0)	1.5708	000010011 000001110 000001101 000000011 100000001 011000011 011000000 110101000 101111000	{104}
2118	9	1/2	2.75	{1, 4, 2, 3}	(0, 0)	0	000001110 000000111 000000111 000000011 000000011 100000001 111000001 111110001 011111110	{1280}
2119	9	1/2	2.75	{6, 1, 2, 4}	(0, 0)	-3.14159	000011111 000001111 000000111 000000011 100000001 110000001 111000001 111100000 111111100	{234}
2120	9	1/2	2.75	{6, 0, 5, 4}	(0, 0)	-1.5708	000010011 000001110 000001101 000000011 100000001 011000011 011000000 110101000 101111000	{104}
2121	9	1/2	2.75	{6, 0, 4, 5}	(1.5708, 0.785398)	-3.14159	000010011 000001110 000001101 000000011 100000001 011000011 011000000 110101000 101111000	{104}
2122	9	1/2	2.75	{1, 4, 3, 2}	(1.5708, 0)	-3.14159	000001110 000000111 000000111 000000011 000000011 100000001 111000001 111110001 011111110	{1280}
2123	9	1/2	2.75	{1, 6, 2, 4}	(1.5708, -1.5708)	3.14159	000011111 000001111 000000111 000000011 100000001 110000001 111000001 111100000 111111100	{234}
2124	9	1/2	2.75	{0, 6, 5, 4}	(1.5708, -0.785398)	3.14159	000010011 000001110 000001101 000000011 100000001 011000011 011000000 110101000 101111000	{104}
2125	9	1/2	2.7	{0, 5, 3, 6}	(1.5708, 0)	-0.927295	000101111 000010011 000010011 100001110 011001111 100110101 100111000 111110001 111011010	{4}
2126	9	1/2	2.7	{0, 5, 6, 3}	(1.5708, -3.14159)	-2.2143	000101111 000010011 000010011 100001110 011001111 100110101 100111000 111110001 111011010	{4}
2127	9	1/2	2.66667	{7, 0, 2, 1}	(0, 0)	1.5708	000000111 000000110 000000011 000000001 000000001 000000001 110000001 111000000 101111100	{8}
2128	9	1/2	2.66667	{1, 4, 2, 3}	(0, 0)	0	000001101 000000111 000000111 000000011 000000011 100000000 111000000 011110000 111110000	{2140}
2129	9	1/2	2.66667	{3, 5, 4, 5}	(0, 0)	-3.14159	000001101 000001001 000000100 000000011 000000011 110000010 101000010 000111101 110110010	{575}
2130	9	1/2	2.66667	{0, 7, 1, 2}	(0, 0)	-1.5708	000000111 000000110 000000011 000000001 000000001 000000001 110000001 111000000 101111100	{8}
2131	9	1/2	2.66667	{0, 7, 2, 1}	(1.5708, 0.785398)	-3.14159	000000111 000000110 000000011 000000001 000000001 000000001 110000001 111000000 101111100	{8}
2132	9	1/2	2.66667	{1, 4, 3, 2}	(1.5708, 0)	-3.14159	000001101 000000111 000000111 000000011 000000011 100000000 111000000 011110000 111110000	{2140}
2133	9	1/2	2.66667	{5, 3, 4, 5}	(1.5708, -1.5708)	3.14159	000001101 000001001 000000100 000000011 000000011 110000010 101000010 000111101 110110010	{575}
2134	9	1/2	2.66667	{7, 0, 1, 2}	(1.5708, -0.785398)	3.14159	000000111 000000110 000000011 000000001 000000001 000000001 110000001 111000000 101111100	{8}
2135	9	1/2	2.63636	{0, 1, 0, 1}	(0, 0)	0	000010011 000010011 000001111 000001101 110000110 001100010 001110001 111011000 111100100	{8}
2136	9	1/2	2.63636	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010011 000010011 000001111 000001101 110000110 001100010 001110001 111011000 111100100	{8}
2137	9	1/2	2.6	{2, 7, 3, 4}	(0.61548, -2.35619)	-1.77215	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2138	9	1/2	2.6	{7, 2, 4, 3}	(0.61548, -0.785398)	1.77215	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2139	9	1/2	2.6	{3, 4, 2, 7}	(0.61548, 2.35619)	-1.77215	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2140	9	1/2	2.6	{4, 3, 7, 2}	(0.61548, 0.785398)	1.77215	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2141	9	1/2	2.6	{7, 2, 3, 4}	(1.23096, 2.35619)	2.49809	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2142	9	1/2	2.6	{3, 4, 7, 2}	(1.23096, -2.35619)	2.49809	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2143	9	1/2	2.6	{4, 3, 2, 7}	(1.23096, -0.785398)	-2.49809	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2144	9	1/2	2.6	{2, 7, 4, 3}	(1.23096, 0.785398)	-2.49809	000101001 000010110 000001101 100000010 010000101 101000011 011010011 010101101 101011110	{3}
2145	9	1/2	2.5	{0, 3, 5, 4}	(1.5708, 0)	-0.927295	000100110 000011111 000011111 100000101 011001010 011010001 111100011 111010101 011101110	{4}
2146	9	1/2	2.5	{0, 3, 4, 5}	(1.5708, -3.14159)	-2.2143	000100110 000011111 000011111 100000101 011001010 011010001 111100011 111010101 011101110	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2147	9	1/2	2.42857	{2, 3, 2, 3}	(0, 0)	0	000001110 000001001 000000111 000000111 000000001 110000010 101100001 101101000 011110100	{48}
2148	9	1/2	2.42857	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001110 000001001 000000111 000000111 000000001 110000010 101100001 101101000 011110100	{48}
2149	9	1/2	2.4	{4, 1, 0, 3}	(0.61548, 2.35619)	1.77215	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2150	9	1/2	2.4	{3, 1, 5, 0}	(0.61548, -2.35619)	-1.77215	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2151	9	1/2	2.4	{0, 3, 4, 1}	(0.61548, -2.35619)	1.77215	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2152	9	1/2	2.4	{3, 0, 1, 4}	(0.61548, -0.785398)	-1.77215	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2153	9	1/2	2.4	{0, 5, 6, 4}	(0.463648, -1.5708)	3.14159	000101111 000011111 000011111 100000011 011000101 111000111 111011000 111101001 111111010	{2}
2154	9	1/2	2.4	{1, 4, 3, 0}	(0.61548, 0.785398)	-1.77215	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2155	9	1/2	2.4	{1, 3, 0, 5}	(0.61548, -0.785398)	1.77215	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2156	9	1/2	2.4	{5, 0, 3, 1}	(0.61548, 2.35619)	-1.77215	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2157	9	1/2	2.4	{4, 6, 5, 0}	(0.463648, 1.5708)	3.14159	000101111 000011111 000011111 100000011 011000101 111000111 111011000 111101001 111111010	{2}
2158	9	1/2	2.4	{0, 5, 1, 3}	(0.61548, 0.785398)	1.77215	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2159	9	1/2	2.4	{1, 3, 5, 0}	(1.23096, 2.35619)	2.49809	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2160	9	1/2	2.4	{1, 4, 0, 3}	(1.23096, -2.35619)	-2.49809	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2161	9	1/2	2.4	{5, 0, 1, 3}	(1.23096, -2.35619)	2.49809	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2162	9	1/2	2.4	{0, 5, 3, 1}	(1.23096, -0.785398)	-2.49809	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2163	9	1/2	2.4	{4, 6, 0, 5}	(1.10715, -1.5708)	3.14159	000101111 000011111 000011111 100000011 011000101 111000111 111011000 111101001 111111010	{2}
2164	9	1/2	2.4	{3, 1, 0, 5}	(1.23096, 0.785398)	-2.49809	000011111 000011011 000001101 000000111 110001111 111010010 101110011 110111101 111110110	{3}
2165	9	1/2	2.4	{4, 1, 3, 0}	(1.23096, -0.785398)	2.49809	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2166	9	1/2	2.4	{0, 3, 1, 4}	(1.23096, 2.35619)	-2.49809	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2167	9	1/2	2.4	{0, 5, 4, 6}	(1.10715, 1.5708)	3.14159	000101111 000011111 000011111 100000011 011000101 111000111 111011000 111101001 111111010	{2}
2168	9	1/2	2.4	{3, 0, 4, 1}	(1.23096, 0.785398)	2.49809	000010011 000001101 000001011 000000101 100000001 011000011 010100001 101001000 111111100	{6}
2169	9	1/2	2.25	{0, 1, 0, 1}	(0, 0)	0	000000111 000000111 000000011 000000011 000000011 000000011 110000001 111111000 111111100	{128}
2170	9	1/2	2.25	{0, 7, 1, 7}	(0, 0)	-3.14159	000000101 000000101 000000011 000000011 000000011 000000011 110000011 001111100 111111100	{16}
2171	9	1/2	2.25	{2, 0, 3, 4}	(0.955317, -0.785398)	-2.0944	000011010 000001001 000000110 000000101 100000011 110000001 001100011 101010101 010111110	{13}
2172	9	1/2	2.25	{0, 2, 4, 3}	(0.955317, -2.35619)	2.0944	000011010 000001001 000000110 000000101 100000011 110000001 001100011 101010101 010111110	{11}
2173	9	1/2	2.25	{3, 4, 2, 0}	(0.955317, 0.785398)	-2.0944	000011010 000001001 000000110 000000101 100000011 110000001 001100011 101010101 010111110	{13}
2174	9	1/2	2.25	{2, 6, 3, 5}	(0.785398, 0)	-3.14159	000011001 000010110 000001011 000000111 110000011 101000010 010100001 011111000 101110100	{8}
2175	9	1/2	2.25	{6, 2, 3, 5}	(1.5708, -1.5708)	1.5708	000011001 000010110 000001011 000000111 110000011 101000010 010100001 011111000 101110100	{8}
2176	9	1/2	2.25	{4, 3, 2, 0}	(0.955317, -0.785398)	2.0944	000011010 000001001 000000110 000000101 100000011 110000001 001100011 101010101 010111110	{13}
2177	9	1/2	2.25	{4, 3, 0, 2}	(0.955317, 2.35619)	2.0944	000011010 000001001 000000110 000000101 100000011 110000001 001100011 101010101 010111110	{11}
2178	9	1/2	2.25	{2, 6, 5, 3}	(1.5708, 1.5708)	1.5708	000011001 000010110 000001011 000000111 110000011 101000010 010100001 011111000 101110100	{8}
2179	9	1/2	2.25	{3, 0, 1, 2}	(0.785398, -3.14159)	-3.14159	000010111 000010100 000001111 000001010 110000101 001100011 111010001 101101001 101011110	{10}
2180	9	1/2	2.25	{7, 1, 0, 2}	(0.955317, -2.35619)	-2.0944	000010110 000010011 000001100 000001011 110000101 001100001 101010001 110100001 010111110	{2}
2181	9	1/2	2.25	{2, 0, 4, 3}	(0.955317, 0.785398)	2.0944	000011010 000001001 000000110 000000101 100000011 110000001 001100011 101010101 010111110	{13}
2182	9	1/2	2.25	{0, 2, 7, 1}	(0.955317, 2.35619)	-2.0944	000010110 000010011 000001100 000001011 110000101 001100001 101010001 110100001 010111110	{2}
2183	9	1/2	2.25	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000111 000000111 000000011 000000011 000000011 000000011 110000001 111111000 111111100	{128}
2184	9	1/2	2.25	{7, 0, 1, 7}	(1.5708, -1.5708)	3.14159	000000101 000000101 000000011 000000011 000000011 000000011 110000011 001111100 111111100	{16}
2185	9	1/2	2.2	{0, 7, 5, 6}	(0, 0)	1.5708	000001001 000000111 000000111 000000110 000000001 100000001 011100001 011100000 111011100	{12}
2186	9	1/2	2.2	{1, 4, 2, 3}	(0, 0)	0	000001100 000000110 000000110 000000011 000000011 100000001 111000001 011110000 000111100	{416}
2187	9	1/2	2.2	{2, 4, 3, 4}	(0, 0)	-3.14159	000010100 000001010 000000111 000000111 100000001 010000001 101100001 011100001 001111110	{146}
2188	9	1/2	2.2	{7, 0, 6, 5}	(0, 0)	-1.5708	000001001 000000111 000000111 000000110 000000001 100000001 011100001 011100000 111011100	{12}
2189	9	1/2	2.2	{4, 1, 6, 3}	(0.463648, 3.14159)	3.14159	000011010 000010011 000001101 000000111 110000110 101000000 001110001 110110001 011100110	{2}
2190	9	1/2	2.2	{1, 3, 5, 2}	(0.61548, 2.35619)	1.77215	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2191	9	1/2	2.2	{1, 6, 2, 3}	(0.61548, -2.35619)	-1.77215	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2192	9	1/2	2.2	{5, 2, 1, 3}	(0.61548, -2.35619)	1.77215	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2193	9	1/2	2.2	{2, 5, 3, 1}	(0.61548, -0.785398)	-1.77215	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2194	9	1/2	2.2	{3, 1, 2, 5}	(0.61548, 0.785398)	-1.77215	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2195	9	1/2	2.2	{1, 4, 3, 6}	(0.463648, 0)	3.14159	000011010 000010011 000001101 000000111 110000110 101000000 001110001 110110001 011100110	{2}
2196	9	1/2	2.2	{6, 1, 3, 2}	(0.61548, -0.785398)	1.77215	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2197	9	1/2	2.2	{2, 3, 1, 6}	(0.61548, 2.35619)	-1.77215	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2198	9	1/2	2.2	{3, 2, 6, 1}	(0.61548, 0.785398)	1.77215	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2199	9	1/2	2.2	{6, 1, 2, 3}	(1.23096, 2.35619)	2.49809	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2200	9	1/2	2.2	{1, 4, 6, 3}	(1.5708, -1.5708)	-2.2143	000011010 000010011 000001101 000000111 110000110 101000000 001110001 110110001 011100110	{2}
2201	9	1/2	2.2	{3, 1, 5, 2}	(1.23096, -2.35619)	-2.49809	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2202	9	1/2	2.2	{2, 3, 6, 1}	(1.23096, -2.35619)	2.49809	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2203	9	1/2	2.2	{3, 2, 1, 6}	(1.23096, -0.785398)	-2.49809	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2204	9	1/2	2.2	{4, 1, 3, 6}	(1.5708, 1.5708)	-2.2143	000011010 000010011 000001101 000000111 110000110 101000000 001110001 110110001 011100110	{2}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2205	9	1/2	2.2	{1, 6, 3, 2}	(1.23096, 0.785398)	-2.49809	000010011 000001111 000001101 000000110 100000010 011000001 011100000 110110001 111001010	{7}
2206	9	1/2	2.2	{1, 3, 2, 5}	(1.23096, -0.785398)	2.49809	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2207	9	1/2	2.2	{5, 2, 3, 1}	(1.23096, 2.35619)	-2.49809	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2208	9	1/2	2.2	{2, 5, 1, 3}	(1.23096, 0.785398)	2.49809	000010101 000001010 000000111 000000110 100000001 010000001 101100011 011100101 101011110	{3}
2209	9	1/2	2.2	{7, 0, 5, 6}	(1.5708, 0.785398)	-3.14159	000001001 000000111 000000111 000000110 000000001 100000001 011100001 011100000 111011100	{12}
2210	9	1/2	2.2	{1, 4, 3, 2}	(1.5708, 0)	-3.14159	000001100 000000110 000000110 000000011 000000011 100000001 111000001 011110000 000111100	{416}
2211	9	1/2	2.2	{4, 2, 3, 4}	(1.5708, -1.5708)	3.14159	000010100 000001010 000001111 000000111 100000001 010000001 101100001 011100001 001111110	{146}
2212	9	1/2	2.2	{0, 7, 6, 5}	(1.5708, -0.785398)	3.14159	000001001 000000111 000000111 000000110 000000001 100000001 011100001 011100000 111011100	{12}
2213	9	1/2	2.14286	{0, 3, 1, 2}	(0, 0)	0	000001011 000001010 000000111 000000110 000000001 110000001 001100001 111100000 101011100	{32}
2214	9	1/2	2.14286	{0, 3, 2, 1}	(1.5708, 0)	-3.14159	000001011 000001010 000000111 000000110 000000001 110000001 001100001 111100000 101011100	{32}
2215	9	1/2	2	{8, 4, 7, 1}	(0.61548, -2.35619)	-1.77215	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2216	9	1/2	2	{4, 8, 1, 7}	(0.61548, -0.785398)	1.77215	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2217	9	1/2	2	{7, 1, 8, 4}	(0.61548, 2.35619)	-1.77215	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2218	9	1/2	2	{1, 7, 4, 8}	(0.61548, 0.785398)	1.77215	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2219	9	1/2	2	{4, 8, 7, 1}	(1.23096, 2.35619)	2.49809	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2220	9	1/2	2	{7, 1, 4, 8}	(1.23096, -2.35619)	2.49809	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2221	9	1/2	2	{1, 7, 8, 4}	(1.23096, -0.785398)	-2.49809	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2222	9	1/2	2	{8, 4, 1, 7}	(1.23096, 0.785398)	-2.49809	000100100 000010100 000001011 100000100 010000011 001000011 110100001 001011000 001011100	{1}
2223	9	1/2	19	{2, 3, 2, 3}	(0, 0)	0	000010110 000010101 000001011 000001011 110000111 001100101 110011001 101110000 011111100	{8}
2224	9	1/2	19	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010110 000010101 000001011 000001011 110000111 001100101 110011001 101110000 011111100	{8}
2225	9	1/2	18	{2, 3, 2, 3}	(0, 0)	0	000011011 000010110 000001101 000001101 110000011 101100011 011100001 110011000 101111100	{24}
2226	9	1/2	18	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000010110 000001101 000001101 110000011 101100011 011100001 110011000 101111100	{24}
2227	9	1/2	15	{0, 6, 3, 2}	(0, 0)	1.5708	000011101 000010011 000001111 000000111 110000110 101000011 101110001 011111000 111101100	{12}
2228	9	1/2	15	{2, 3, 2, 3}	(0, 0)	0	000010101 000010010 000001111 000001111 110000111 001100001 101110011 011110100 101111100	{32}
2229	9	1/2	15	{6, 0, 2, 3}	(0, 0)	-1.5708	000011101 000010011 000001111 000000111 110000110 101000011 101110001 011111000 111101100	{12}
2230	9	1/2	15	{6, 0, 3, 2}	(1.5708, 0.785398)	-3.14159	000011101 000010011 000001111 000000111 110000110 101000011 101110001 011111000 111101100	{12}
2231	9	1/2	15	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000010010 000001111 000001111 110000111 001100001 101110011 011110100 101111100	{32}
2232	9	1/2	15	{0, 6, 2, 3}	(1.5708, -0.785398)	3.14159	000011101 000010011 000001111 000000111 110000110 101000011 101110001 011111000 111101100	{12}
2233	9	1/2	14	{3, 8, 7, 6}	(0, 0)	1.5708	000011011 000010011 000001101 000000111 110000101 101000011 001110000 110101000 111111000	{16}
2234	9	1/2	14	{8, 3, 6, 7}	(0, 0)	-1.5708	000011011 000010011 000001101 000000111 110000101 101000011 001110000 110101000 111111000	{16}
2235	9	1/2	14	{8, 3, 7, 6}	(1.5708, 0.785398)	-3.14159	000011011 000010011 000001101 000000111 110000101 101000011 001110000 110101000 111111000	{16}
2236	9	1/2	14	{3, 8, 6, 7}	(1.5708, -0.785398)	3.14159	000011011 000010011 000001101 000000111 110000101 101000011 001110000 110101000 111111000	{16}
2237	9	1/2	13	{3, 6, 8, 4}	(0, 0)	1.5708	000010110 000001111 000001101 000000011 100000011 011000010 111000001 110111000 011110100	{20}
2238	9	1/2	13	{6, 3, 4, 8}	(0, 0)	-1.5708	000010110 000001111 000001101 000000011 100000011 011000010 111000001 110111000 011110100	{20}
2239	9	1/2	13	{6, 3, 8, 4}	(1.5708, 0.785398)	-3.14159	000010110 000001111 000001101 000000011 100000011 011000010 111000001 110111000 011110100	{20}
2240	9	1/2	13	{3, 6, 4, 8}	(1.5708, -0.785398)	3.14159	000010110 000001111 000001101 000000011 100000011 011000010 111000001 110111000 011110100	{20}
2241	9	1/2	11	{1, 7, 1, 3}	(0, 0)	1.5708	000011011 000001100 000000111 000000010 100000111 110000001 011010001 101110000 101011100	{248}
2242	9	1/2	11	{7, 1, 3, 1}	(0, 0)	-1.5708	000011011 000001100 000000111 000000010 100000111 110000001 011010001 101110000 101011100	{248}
2243	9	1/2	11	{7, 1, 1, 3}	(1.5708, 0.785398)	-3.14159	000011011 000001100 000000111 000000010 100000111 110000001 011010001 101110000 101011100	{248}
2244	9	1/2	11	{1, 7, 3, 1}	(1.5708, -0.785398)	3.14159	000011011 000001100 000000111 000000010 100000111 110000001 011010001 101110000 101011100	{248}
2245	9	1/2	10.5	{1, 2, 1, 2}	(0, 0)	0	000011101 000001110 000001110 000000011 100000011 111000001 111000001 011110001 100111110	{112}
2246	9	1/2	10.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001110 000001110 000000011 100000011 111000001 111000001 011110001 100111110	{112}
2247	9	1/2	10	{1, 7, 3, 2}	(0, 0)	1.5708	000011001 000001110 000000111 000000011 100000000 110000011 011000001 011101000 101101100	{200}
2248	9	1/2	10	{0, 4, 7, 4}	(0, 0)	-3.14159	000101111 000011111 000011011 100000110 011000101 111000011 110110001 111101000 111011100	{17}
2249	9	1/2	10	{7, 1, 2, 3}	(0, 0)	-1.5708	000011001 000001110 000000111 000000011 100000000 110000011 011000001 011101000 101101100	{200}
2250	9	1/2	10	{7, 1, 3, 2}	(1.5708, 0.785398)	-3.14159	000011001 000001110 000000111 000000011 100000000 110000011 011000001 011101000 101101100	{200}
2251	9	1/2	10	{4, 0, 7, 4}	(1.5708, -1.5708)	3.14159	000101111 000011111 000011011 100000110 011000101 111000011 110110001 111101000 111011100	{17}
2252	9	1/2	10	{1, 7, 2, 3}	(1.5708, -0.785398)	3.14159	000011001 000001110 000000111 000000011 100000000 110000011 011000001 011101000 101101100	{200}
2253	9	1/2	1.90909	{2, 3, 2, 3}	(0, 0)	0	000011101 000001011 000000110 000000110 100001011 110010000 101100001 011110001 110010110	{8}
2254	9	1/2	1.90909	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000001011 000000110 000000110 100001011 110010000 101100001 011110001 110010110	{8}
2255	9	1/2	1.9	{4, 6, 5, 7}	(0.463648, 3.14159)	3.14159	000101111 000010111 000010111 100001110 011001101 100110010 111110001 111101001 111010110	{2}
2256	9	1/2	1.9	{7, 2, 6, 3}	(1.5708, -1.5708)	-0.927295	000010111 000010111 000001111 000001001 110000110 001100011 111010001 111011001 111101110	{4}
2257	9	1/2	1.9	{0, 1, 4, 5}	(1.5708, 0)	-0.927295	000010011 000001011 000000111 000000111 100000001 010000001 001100011 111100101 111111110	{24}
2258	9	1/2	1.9	{2, 7, 3, 6}	(1.5708, 1.5708)	-0.927295	000010111 000010111 000001111 000001001 110000110 001100011 111010001 111011001 111101110	{4}
2259	9	1/2	1.9	{6, 4, 7, 5}	(0.463648, 0)	3.14159	000101111 000010111 000010111 100001110 011001101 100110010 111110001 111101001 111010110	{2}
2260	9	1/2	1.9	{0, 5, 1, 4}	(1.5708, -3.14159)	-0.927295	000011011 000011001 000000111 000000111 110000011 110000001 001100010 101110101 111111010	{8}
2261	9	1/2	1.9	{2, 7, 6, 3}	(1.10715, 3.14159)	3.14159	000010111 000010111 000001111 000001001 110000110 001100011 111010001 111011001 111101110	{4}
2262	9	1/2	1.9	{6, 4, 5, 7}	(1.5708, -1.5708)	-2.2143	000101111 000010111 000010111 100001110 011001101 100110010 111110001 111101001 111010110	{2}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2263	9	1/2	1.9	{0, 5, 4, 1}	(1.5708, 0)	-2.2143	000011011 000011001 000000111 000000111 110000011 110000001 001100010 101110101 111111010	{8}
2264	9	1/2	1.9	{4, 6, 7, 5}	(1.5708, 1.5708)	-2.2143	000101111 000010111 000010111 100001110 011001101 100110010 111110001 111101001 111010110	{2}
2265	9	1/2	1.9	{7, 2, 3, 6}	(1.10715, 0)	3.14159	000010111 000010111 000001111 000001001 110000110 001100011 111010001 111011001 111101110	{4}
2266	9	1/2	1.9	{0, 1, 5, 4}	(1.5708, -3.14159)	-2.2143	000010011 000001011 000000111 000000111 100000001 010000001 001100011 111100101 111111110	{24}
2267	9	1/2	1.875	{2, 5, 3, 4}	(0, 0)	0	000000110 000000110 000000101 000000101 000000011 000000011 111100001 110011001 001111110	{432}
2268	9	1/2	1.875	{0, 4, 1, 4}	(0, 0)	-3.14159	000001011 000001011 000000110 000000110 000000101 110000101 001111001 111100001 110011110	{110}
2269	9	1/2	1.875	{2, 5, 4, 3}	(1.5708, 0)	-3.14159	000000110 000000110 000000101 000000101 000000011 000000011 111100001 110011001 001111110	{432}
2270	9	1/2	1.875	{4, 0, 1, 4}	(1.5708, -1.5708)	3.14159	000001011 000001011 000000110 000000110 000000101 110000101 001111001 111100001 110011110	{110}
2271	9	1/2	1.85714	{0, 2, 5, 1}	(0, 0)	1.5708	000001011 000000101 000000101 000000011 000000011 100000011 011000011 100111100 111111100	{31}
2272	9	1/2	1.85714	{2, 4, 3, 4}	(0, 0)	-3.14159	000010111 000010111 000001011 000001011 110001110 001110111 110011001 111111001 111101110	{30}
2273	9	1/2	1.85714	{2, 0, 1, 5}	(0, 0)	-1.5708	000001011 000000101 000000101 000000011 000000011 100000011 011000011 100111100 111111100	{31}
2274	9	1/2	1.85714	{2, 0, 5, 1}	(1.5708, 0.785398)	-3.14159	000001011 000000101 000000101 000000011 000000011 100000011 011000011 100111100 111111100	{31}
2275	9	1/2	1.85714	{4, 2, 3, 4}	(1.5708, -1.5708)	3.14159	000010111 000010111 000001011 000001011 110001110 001110111 110011001 111111001 111101110	{30}
2276	9	1/2	1.85714	{0, 2, 1, 5}	(1.5708, -0.785398)	3.14159	000001011 000000101 000000101 000000011 000000011 100000011 011000011 100111100 111111100	{31}
2277	9	1/2	1.83333	{2, 3, 2, 3}	(0, 0)	0	000001111 000001010 000000101 000000101 000000011 110000001 101100001 110010000 101111100	{120}
2278	9	1/2	1.83333	{5, 7, 6, 7}	(0, 0)	-3.14159	000001111 000001111 000000011 000000011 000000011 110000001 110000001 111110000 111111100	{16}
2279	9	1/2	1.83333	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000101 000000011 110000001 101100001 110010000 101111100	{120}
2280	9	1/2	1.83333	{7, 5, 6, 7}	(1.5708, -1.5708)	3.14159	000001111 000001111 000000011 000000011 000000011 110000001 110000001 111110000 111111100	{16}
2281	9	1/2	1.8	{3, 4, 3, 4}	(0, 0)	0	000001010 000000111 000000100 000000011 000000011 100000001 011000001 110110001 010111110	{120}
2282	9	1/2	1.8	{2, 4, 3, 4}	(0, 0)	-3.14159	000011011 000001001 000000111 000000111 100000001 110000000 001100000 101100001 111110010	{20}
2283	9	1/2	1.8	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001010 000000111 000000100 000000011 000000011 100000001 011000001 110110001 010111110	{120}
2284	9	1/2	1.8	{4, 2, 3, 4}	(1.5708, -1.5708)	3.14159	000011011 000001001 000000111 000000111 100000001 110000000 001100000 101100001 111110010	{20}
2285	9	1/2	1.75	{8, 2, 4, 3}	(0, 0)	1.5708	000001101 000001011 000000111 000000111 000000001 110000000 101100000 011100000 111110000	{190}
2286	9	1/2	1.75	{2, 8, 3, 4}	(0, 0)	-1.5708	000001101 000001011 000000111 000000111 000000001 110000000 101100000 011100000 111110000	{190}
2287	9	1/2	1.75	{2, 8, 4, 3}	(1.5708, 0.785398)	-3.14159	000001101 000001011 000000111 000000111 000000001 110000000 101100000 011100000 111110000	{190}
2288	9	1/2	1.75	{8, 2, 3, 4}	(1.5708, -0.785398)	3.14159	000001101 000001011 000000111 000000111 000000001 110000000 101100000 011100000 111110000	{190}
2289	9	1/2	1.72727	{2, 3, 2, 3}	(0, 0)	0	000011100 000001111 000000011 000000011 100000110 110000001 110010001 011110000 011101100	{80}
2290	9	1/2	1.72727	{3, 0, 2, 0}	(0, 0)	-3.14159	000011100 000001010 000000111 000000111 100001011 110010001 101100001 011110000 001111100	{14}
2291	9	1/2	1.72727	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011100 000001111 000000011 000000011 100000110 110000001 110010001 011110000 011101100	{80}
2292	9	1/2	1.72727	{0, 3, 2, 0}	(1.5708, -1.5708)	3.14159	000011100 000001010 000000111 000000111 100001011 110010001 101100001 011110000 001111100	{14}
2293	9	1/2	1.71429	{1, 4, 2, 3}	(0, 0)	0	000001110 000000101 000000101 000000011 000000011 100000000 111000000 100110000 011110000	{504}
2294	9	1/2	1.71429	{2, 0, 1, 0}	(0, 0)	-3.14159	000010110 000001101 000001101 000000010 100000011 011000010 111000001 100111001 011010110	{148}
2295	9	1/2	1.71429	{1, 4, 3, 2}	(1.5708, 0)	-3.14159	000001110 000000101 000000101 000000011 000000011 100000000 111000000 100110000 011110000	{504}
2296	9	1/2	1.71429	{0, 2, 1, 0}	(1.5708, -1.5708)	3.14159	000010110 000001101 000001101 000000010 100000011 011000010 111000001 100111001 011010110	{148}
2297	9	1/2	1.7	{2, 3, 5, 4}	(1.5708, 0)	-0.927295	000011111 000011111 000010110 000001110 111000001 110100001 111100011 111100101 110011110	{4}
2298	9	1/2	1.7	{2, 3, 4, 5}	(1.5708, -3.14159)	-2.2143	000011111 000011111 000010110 000001110 111000001 110100001 111100011 111100101 110011110	{4}
2299	9	1/2	1.66667	{7, 1, 6, 2}	(1.5708, -1.5708)	-0.927295	000100111 000011111 000011001 100000110 011001011 011010000 110100001 110110001 111010110	{2}
2300	9	1/2	1.66667	{1, 7, 2, 6}	(1.5708, 1.5708)	-0.927295	000100111 000011111 000011001 100000110 011001011 011010000 110100001 110110001 111010110	{2}
2301	9	1/2	1.66667	{1, 5, 2, 4}	(1.5708, -3.14159)	-0.927295	000100101 000011111 000011011 100000100 011000111 011000011 110110001 011011000 111011100	{4}
2302	9	1/2	1.66667	{1, 7, 6, 2}	(1.10715, 3.14159)	3.14159	000100111 000011111 000011001 100000110 011001011 011010000 110100001 110110001 111010110	{2}
2303	9	1/2	1.66667	{1, 5, 4, 2}	(1.5708, 0)	-2.2143	000100101 000011111 000011011 100000100 011000111 011000011 110110001 011011000 111011100	{4}
2304	9	1/2	1.66667	{7, 1, 2, 6}	(1.10715, 0)	3.14159	000100111 000011111 000011001 100000110 011001011 011010000 110100001 110110001 111010110	{2}
2305	9	1/2	1.625	{3, 4, 3, 4}	(0, 0)	0	000001001 000001001 000000111 000000110 000000110 110000010 001110001 001111000 111000100	{424}
2306	9	1/2	1.625	{0, 2, 1, 2}	(0, 0)	-3.14159	000011010 000011010 000000111 000000110 110000001 110000001 001100001 111100000 001011100	{166}
2307	9	1/2	1.625	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000001001 000000111 000000110 000000110 110000010 001110001 001111000 111000100	{424}
2308	9	1/2	1.625	{2, 0, 1, 2}	(1.5708, -1.5708)	3.14159	000011010 000011010 000000111 000000110 110000001 110000001 001100001 111100000 001011100	{166}
2309	9	1/2	1.6	{0, 7, 5, 6}	(0, 0)	1.5708	000001110 000000111 000000110 000000001 000000001 100000000 111000001 111000001 010110110	{64}
2310	9	1/2	1.6	{7, 0, 6, 5}	(0, 0)	-1.5708	000001110 000000111 000000110 000000001 000000001 100000000 111000001 111000001 010110110	{64}
2311	9	1/2	1.6	{7, 0, 5, 6}	(1.5708, 0.785398)	-3.14159	000001110 000000111 000000110 000000001 000000001 100000000 111000001 111000001 010110110	{64}
2312	9	1/2	1.6	{0, 7, 6, 5}	(1.5708, -0.785398)	3.14159	000001110 000000111 000000110 000000001 000000001 100000000 111000001 111000001 010110110	{64}
2313	9	1/2	1.57143	{6, 7, 6, 7}	(0, 0)	0	000001111 000001000 000000111 000000111 000000111 110000001 101110000 101110000 101111000	{32}
2314	9	1/2	1.57143	{5, 7, 6, 7}	(0, 0)	-3.14159	000001111 000001100 000000011 000000011 000000011 110000001 110000001 101110000 101111100	{16}
2315	9	1/2	1.57143	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000001111 000001000 000000111 000000111 000000111 110000001 101110000 101110000 101111000	{32}
2316	9	1/2	1.57143	{7, 5, 6, 7}	(1.5708, -1.5708)	3.14159	000001111 000001100 000000011 000000011 000000011 110000001 110000001 101110000 101111100	{16}
2317	9	1/2	1.54546	{2, 3, 2, 3}	(0, 0)	0	000001100 000001011 000000111 000000111 000000011 110000010 101100001 011111000 011110100	{48}
2318	9	1/2	1.54546	{1, 4, 2, 4}	(0, 0)	-3.14159	000010110 000001101 000001101 000000011 100000011 011000011 111000000 100111000 011111000	{34}
2319	9	1/2	1.54546	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000111 000000011 110000010 101100001 011111000 011110100	{48}
2320	9	1/2	1.54546	{4, 1, 2, 4}	(1.5708, -1.5708)	3.14159	000010110 000001101 000001101 000000011 100000011 011000011 111000000 100111000 011111000	{34}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2321	9	1/2	1.42857	{2, 3, 2, 3}	(0, 0)	0	000001010 000000111 000000110 000000110 000000001 100000001 011100001 111100000 010011100	{108}
2322	9	1/2	1.42857	{8, 1, 7, 1}	(0, 0)	-3.14159	000010111 000001111 000001100 000000011 100000011 011000011 111000000 110111000 110111000	{18}
2323	9	1/2	1.42857	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001010 000000111 000000110 000000110 000000001 100000001 011100001 111100000 010011100	{108}
2324	9	1/2	1.42857	{1, 8, 7, 1}	(1.5708, -1.5708)	3.14159	000010111 000001111 000001100 000000011 100000011 011000011 111000000 110111000 110111000	{18}
2325	9	1/2	1.4	{1, 0, 1, 5}	(0, 0)	1.5708	000001001 000000111 000000110 000000001 000000001 100000000 011000001 011000000 110110100	{22}
2326	9	1/2	1.4	{0, 1, 5, 1}	(0, 0)	-1.5708	000001001 000000111 000000110 000000001 000000001 100000000 011000001 011000000 110110100	{22}
2327	9	1/2	1.4	{0, 1, 1, 5}	(1.5708, 0.785398)	-3.14159	000001001 000000111 000000110 000000001 000000001 100000000 011000001 011000000 110110100	{22}
2328	9	1/2	1.4	{1, 0, 5, 1}	(1.5708, -0.785398)	3.14159	000001001 000000111 000000110 000000001 000000001 100000000 011000001 011000000 110110100	{22}
2329	9	1/2	1.375	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001110 000000001 000000001 111000001 111000000 111000000 110111000	{40}
2330	9	1/2	1.375	{5, 6, 7, 6}	(0, 0)	-3.14159	000001111 000001010 000001010 000000101 000000101 111000001 100110000 111000001 100111010	{13}
2331	9	1/2	1.375	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001110 000000001 000000001 111000001 111000000 111000000 110111000	{40}
2332	9	1/2	1.375	{6, 5, 7, 6}	(1.5708, -1.5708)	3.14159	000001111 000001010 000001010 000000101 000000101 111000001 100110000 111000001 100111010	{13}
2333	9	1/2	1.35714	{0, 1, 0, 1}	(0, 0)	0	000011110 000011110 000001111 000001111 110000011 111100001 111100001 111110000 001111100	{8}
2334	9	1/2	1.35714	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011110 000011110 000001111 000001111 110000011 111100001 111100001 111110000 001111100	{8}
2335	9	1/2	1.28571	{1, 2, 1, 2}	(0, 0)	0	000000111 000000110 000000110 000000001 000000001 000000001 111000001 111000000 100111100	{8}
2336	9	1/2	1.28571	{8, 0, 7, 0}	(0, 0)	-3.14159	000001111 000001100 000000011 000000011 000000011 110000011 110000000 101111000 101111000	{8}
2337	9	1/2	1.28571	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000111 000000110 000000110 000000001 000000001 000000001 111000001 111000000 100111100	{8}
2338	9	1/2	1.28571	{0, 8, 7, 0}	(1.5708, -1.5708)	3.14159	000001111 000001100 000000011 000000011 000000011 110000011 110000000 101111000 101111000	{8}
2339	9	1/2	1.25	{1, 2, 1, 2}	(0, 0)	0	000000011 000000010 000000010 000000001 000000001 000000001 000000001 111000000 100111100	{104}
2340	9	1/2	1.25	{8, 0, 7, 0}	(0, 0)	-3.14159	000000111 000000100 000000011 000000011 000000011 000000011 110000000 101111000 101111000	{8}
2341	9	1/2	1.25	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000011 000000010 000000010 000000001 000000001 000000001 000000001 111000000 100111100	{104}
2342	9	1/2	1.25	{0, 8, 7, 0}	(1.5708, -1.5708)	3.14159	000000111 000000100 000000011 000000011 000000011 000000011 110000000 101111000 101111000	{8}
2343	9	1/2	1.2	{0, 8, 6, 7}	(0, 0)	1.5708	000000111 000000011 000000011 000000011 000000011 000000011 100000000 111111000 111111000	{4}
2344	9	1/2	1.2	{3, 4, 3, 4}	(0, 0)	0	000001010 000000111 000000110 000000001 000000001 100000001 011000000 111000000 010111000	{632}
2345	9	1/2	1.2	{8, 0, 7, 6}	(0, 0)	-1.5708	000000111 000000011 000000011 000000011 000000011 000000011 100000000 111111000 111111000	{4}
2346	9	1/2	1.2	{8, 0, 6, 7}	(1.5708, 0.785398)	-3.14159	000000111 000000011 000000011 000000011 000000011 000000011 100000000 111111000 111111000	{4}
2347	9	1/2	1.2	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001010 000000111 000000110 000000001 000000001 100000001 011000000 111000000 010111000	{632}
2348	9	1/2	1.2	{0, 8, 7, 6}	(1.5708, -0.785398)	3.14159	000000111 000000011 000000011 000000011 000000011 000000011 100000000 111111000 111111000	{4}
2349	9	1/2	1.18182	{3, 4, 3, 4}	(0, 0)	0	000001111 000001100 000000111 000000011 000000011 110000001 111000010 101110100 101111000	{16}
2350	9	1/2	1.18182	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001100 000000111 000000011 000000011 110000001 111000010 101110100 101111000	{16}
2351	9	1/2	1.1	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001000 000000111 000000111 111000001 110110000 110110000 110111000	{16}
2352	9	1/2	1.1	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001000 000000111 000000111 111000001 110110000 110110000 110111000	{16}
2353	9	1/2	0.916667	{2, 3, 2, 3}	(0, 0)	0	000001110 000000101 000000011 000000011 000000011 100000101 110001011 101110100 011111100	{624}
2354	9	1/2	0.916667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001110 000000101 000000011 000000011 000000011 100000101 110001011 101110100 011111100	{624}
2355	9	1/2	0.909091	{2, 3, 2, 3}	(0, 0)	0	000001110 000001101 000000011 000000011 000000011 110000001 110000000 101110000 011111000	{336}
2356	9	1/2	0.909091	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001110 000001101 000000011 000000011 000000011 110000001 110000000 101110000 011111000	{336}
2357	9	1/2	0.9	{0, 1, 0, 1}	(0, 0)	0	000000111 000000111 000000100 000000100 000000011 000000011 111100001 110011000 110011100	{8}
2358	9	1/2	0.9	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000111 000000111 000000100 000000100 000000011 000000011 111100001 110011000 110011100	{8}
2359	9	1/2	0.875	{2, 3, 2, 3}	(0, 0)	0	000001111 000000100 000000011 000000011 000000011 100000001 110000010 101110100 101111000	{2000}
2360	9	1/2	0.875	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000000100 000000011 000000011 000000011 100000001 110000010 101110100 101111000	{2000}
2361	9	1/2	0.833333	{0, 1, 0, 1}	(0, 0)	0	000000011 000000011 000000010 000000010 000000001 000000001 000000001 111100000 110011100	{1280}
2362	9	1/2	0.833333	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000011 000000011 000000010 000000010 000000001 000000001 000000001 111100000 110011100	{1280}
2363	9	1/2	0.8	{1, 2, 1, 2}	(0, 0)	0	000000111 000000100 000000100 000000011 000000011 000000011 111000001 100111000 100111100	{2736}
2364	9	1/2	0.8	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000111 000000100 000000100 000000011 000000011 000000011 111000001 100111000 100111100	{2736}
2365	9	1/2	0.727273	{3, 4, 3, 4}	(0, 0)	0	000001110 000001001 000000111 000000001 000000001 110000010 101000000 101001000 011110000	{96}
2366	9	1/2	0.727273	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001110 000001001 000000111 000000001 000000001 110000010 101000000 101001000 011110000	{96}
2367	9	1/2	0.7	{0, 1, 0, 1}	(0, 0)	0	000000111 000000111 000000111 000000011 000000011 000000011 111000001 111111000 111111100	{192}
2368	9	1/2	0.7	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000111 000000111 000000111 000000011 000000011 000000011 111000001 111111000 111111100	{192}
2369	9	1/2	0.65	{2, 3, 2, 3}	(0, 0)	0	000001011 000001011 000000111 000000111 000000111 110000110 001111001 111111000 111110100	{136}
2370	9	1/2	0.65	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000111 000000111 110000110 001111001 111111000 111110100	{136}
2371	9	1/2	0.636364	{2, 3, 2, 3}	(0, 0)	0	000001110 000000101 000000011 000000011 000000011 100000011 110000000 101111000 011111000	{160}
2372	9	1/2	0.636364	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001110 000000101 000000011 000000011 000000011 100000011 110000000 101111000 011111000	{160}
2373	9	1/2	0.615385	{2, 3, 2, 3}	(0, 0)	0	000001111 000001000 000000111 000000111 000000111 110000001 101110000 101110000 101111000	{80}
2374	9	1/2	0.615385	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000001000 000000111 000000111 000000111 110000001 101110000 101110000 101111000	{80}
2375	9	1/2	0.583333	{3, 4, 3, 4}	(0, 0)	0	000000111 000000100 000000100 000000011 000000011 000000011 111000001 100111000 100111100	{88}
2376	9	1/2	0.583333	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000000111 000000100 000000100 000000011 000000011 000000011 111000001 100111000 100111100	{88}
2377	9	1/2	0.571429	{1, 2, 1, 2}	(0, 0)	0	000000011 000000010 000000010 000000010 000000001 000000001 000000001 111100000 100011100	{160}
2378	9	1/2	0.571429	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000011 000000010 000000010 000000010 000000001 000000001 000000001 111100000 100011100	{160}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2379	9	1/2	0.545455	{3, 4, 3, 4}	(0, 0)	0	000001010 000000111 000000110 000000001 000000001 100000001 011000001 111000000 010111100	{16}
2380	9	1/2	0.545455	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001010 000000111 000000110 000000001 000000001 100000001 011000001 111000000 010111100	{16}
2381	9	1/2	0.538462	{3, 4, 3, 4}	(0, 0)	0	000000111 000000111 000000100 000000011 000000011 000000011 111000001 110111000 110111100	{24}
2382	9	1/2	0.538462	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000000111 000000111 000000100 000000011 000000011 000000011 111000001 110111000 110111100	{24}
2383	9	1/2	0.461538	{3, 4, 3, 4}	(0, 0)	0	000000111 000000110 000000110 000000001 000000001 000000001 111000001 111000000 100111100	{24}
2384	9	1/2	0.461538	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000000111 000000110 000000110 000000001 000000001 000000001 111000001 111000000 100111100	{24}
2385	9	1/2	0.416667	{2, 3, 2, 3}	(0, 0)	0	000000111 000000111 000000011 000000011 000000011 000000011 110000001 111111000 111111100	{288}
2386	9	1/2	0.416667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000111 000000111 000000011 000000011 000000011 000000011 110000001 111111000 111111100	{288}
2387	9	1/2	0.384615	{2, 3, 2, 3}	(0, 0)	0	000000111 000000100 000000011 000000011 000000011 000000011 110000001 101111000 101111100	{48}
2388	9	1/2	0.384615	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000111 000000100 000000011 000000011 000000011 000000011 110000001 101111000 101111100	{48}
2389	9	1/2	0.375	{3, 4, 3, 4}	(0, 0)	0	000000011 000000010 000000010 000000001 000000001 000000001 000000001 111000000 100111100	{288}
2390	9	1/2	0.375	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000000011 000000010 000000010 000000001 000000001 000000001 000000001 111000000 100111100	{288}
2391	9	1/2	0.3	{1, 2, 1, 2}	(0, 0)	0	000000111 000000011 000000011 000000011 000000011 000000011 100000001 111111000 111111100	{80}
2392	9	1/2	0.3	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000111 000000011 000000011 000000011 000000011 000000011 100000001 111111000 111111100	{80}
2393	9	1/2	0.285714	{2, 3, 2, 3}	(0, 0)	0	000000011 000000010 000000001 000000001 000000001 000000001 000000001 110000000 101111100	{160}
2394	9	1/2	0.285714	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000011 000000010 000000001 000000001 000000001 000000001 000000001 110000000 101111100	{160}
2395	9	1/2	0.166667	{0, 1, 0, 1}	(0, 0)	0	000000001 000000001 000000001 000000001 000000001 000000001 000000001 000000001 111111110	{112}
2396	9	1/2	0.166667	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000001 000000001 000000001 000000001 000000001 000000001 000000001 000000001 111111110	{112}
2397	9	2/5	9.75078	{1, 2, 1, 2}	(0, 0)	0	000010011 000001101 000001101 000000110 100000010 011000111 011101001 100111001 111001110	{8}
2398	9	2/5	9.75078	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001101 000001101 000000110 100000010 011000111 011101001 100111001 111001110	{8}
2399	9	2/5	9.73607	{6, 7, 6, 7}	(0, 0)	0	000011111 000001111 000001110 000000111 100001111 111010111 111111001 111111001 110111110	{16}
2400	9	2/5	9.73607	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000001111 000001110 000000111 100001111 111010111 111111001 111111001 110111110	{16}
2401	9	2/5	9.34752	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{4}
2402	9	2/5	9.34752	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{4}
2403	9	2/5	75.7771	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000011 100000000 011000000 011000000 111100000 111100000	{12}
2404	9	2/5	75.7771	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000011 100000000 011000000 011000000 111100000 111100000	{12}
2405	9	2/5	7.63932	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001000 100000011 011100000 111000001 111010001 111010110	{4}
2406	9	2/5	7.63932	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000001000 100000011 011100000 111000001 111010001 111010110	{4}
2407	9	2/5	68.541	{2, 3, 2, 3}	(0, 0)	0	000011010 000001111 000000011 000000011 100000101 110000000 010010000 111100001 011110010	{16}
2408	9	2/5	68.541	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011010 000001111 000000011 000000011 100000101 110000000 010010000 111100001 011110010	{16}
2409	9	2/5	64.0689	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000110 000000001 110000011 001100001 111101001 111011110	{8}
2410	9	2/5	64.0689	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000110 000000001 110000011 001100001 111101001 111011110	{8}
2411	9	2/5	6.98684	{1, 2, 1, 2}	(0, 0)	0	000101100 000011111 000011111 100000011 011000110 111000001 111010000 011110001 011101010	{8}
2412	9	2/5	6.98684	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101100 000011111 000011111 100000011 011000110 111000001 111010000 011110001 011101010	{8}
2413	9	2/5	6.58359	{2, 3, 2, 3}	(0, 0)	0	000001101 000001100 000000011 000000011 000000010 110000101 110001001 001110000 101101100	{48}
2414	9	2/5	6.58359	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001100 000000011 000000011 000000010 110000101 110001001 001110000 101101100	{48}
2415	9	2/5	6.52786	{3, 8, 7, 4}	(0, 0)	0	000001100 000001011 000000111 000000010 000000001 110000011 101000011 011101100 011011100	{16}
2416	9	2/5	6.52786	{3, 8, 4, 7}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000010 000000001 110000011 101000011 011101100 011011100	{16}
2417	9	2/5	52.3607	{1, 2, 1, 2}	(0, 0)	0	000010101 000001111 000001111 000000011 100000011 011000001 111000010 011110100 111111000	{20}
2418	9	2/5	52.3607	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001111 000001111 000000011 100000011 011000001 111000010 011110100 111111000	{20}
2419	9	2/5	50.5967	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100000 111101100	{8}
2420	9	2/5	50.5967	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100000 111101100	{8}
2421	9	2/5	5.78885	{3, 4, 3, 4}	(0, 0)	0	000001111 000001011 000000110 000000101 000000101 110000010 101110001 111001001 110110110	{20}
2422	9	2/5	5.78885	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001011 000000110 000000101 000000101 110000010 101110001 111001001 110110110	{20}
2423	9	2/5	5.42705	{2, 3, 2, 3}	(0, 0)	0	000001101 000000100 000000010 000000010 000000001 100000000 110000001 001100001 100010110	{32}
2424	9	2/5	5.42705	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000000100 000000010 000000010 000000001 100000000 110000001 001100001 100010110	{32}
2425	9	2/5	5.27864	{1, 2, 1, 2}	(0, 0)	0	000011011 000000111 000000111 000000100 100001011 100010001 011100000 111010001 111011010	{4}
2426	9	2/5	5.27864	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000000111 000000111 000000100 100001011 100010001 011100000 111010001 111011010	{4}
2427	9	2/5	5.26393	{6, 7, 6, 7}	(0, 0)	0	000011111 000001111 000000111 000000110 100000111 110000111 111111001 111111001 111011110	{16}
2428	9	2/5	5.26393	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000001111 000000111 000000110 100000111 110000111 111111001 111111001 111011110	{16}
2429	9	2/5	5.10557	{2, 3, 2, 3}	(0, 0)	0	000010111 000001101 000001011 000001011 100000100 011100011 110010010 101101101 111101010	{4}
2430	9	2/5	5.10557	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001101 000001011 000001011 100000100 011100011 110010010 101101101 111101010	{4}
2431	9	2/5	47.8885	{2, 3, 2, 3}	(0, 0)	0	000011110 000001011 000000111 000000111 100000100 110000001 101110001 111100000 011101100	{4}
2432	9	2/5	47.8885	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000001011 000000111 000000111 100000100 110000001 101110001 111100000 011101100	{4}
2433	9	2/5	45.1246	{2, 3, 2, 3}	(0, 0)	0	000011001 000001011 000000101 000000101 100000010 110000000 001100010 010010100 111100000	{24}
2434	9	2/5	45.1246	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011001 000001011 000000101 000000101 100000010 110000000 001100010 010010100 111100000	{24}
2435	9	2/5	40.6525	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000110 000000101 110000011 001110001 111101001 111011110	{8}
2436	9	2/5	40.6525	{0, 1, 1, 0}	(1.5708, 0)	3.14159	000001011 000001011 000000111 000000110 000000101 110000011 001110001 111101001 111011110	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2437	9	2/5	4.87539	{2, 3, 2, 3}	(0, 0)	0	000011001 000001010 000000101 000000101 100000011 110000000 001100010 010010100 101110000	{24}
2438	9	2/5	4.87539	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011001 000001010 000000101 000000101 100000011 110000000 001100010 010010100 101110000	{24}
2439	9	2/5	4.83688	{7, 8, 7, 8}	(0, 0)	0	000001111 000001111 000001011 000000111 000000011 111000111 110101011 111111100 111111100	{4}
2440	9	2/5	4.83688	{7, 8, 8, 7}	(1.5708, 0)	-3.14159	000001111 000001111 000001011 000000111 000000011 111000111 110101011 111111100 111111100	{4}
2441	9	2/5	4.78346	{0, 2, 5, 6}	(1.5708, 0)	0.96804	000111101 000101111 000011111 110010101 101100011 111000011 111100001 011011001 111111110	{4}
2442	9	2/5	4.78346	{0, 2, 6, 5}	(1.5708, 0)	-2.17355	000111101 000101111 000011111 110010101 101100011 111000011 111100001 011011001 111111110	{4}
2443	9	2/5	4.22291	{7, 8, 7, 8}	(0, 0)	0	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{32}
2444	9	2/5	4.22291	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{32}
2445	9	2/5	4	{1, 2, 1, 2}	(0, 0)	0	000010010 000001111 000001111 000001100 100000010 011100101 011101001 111010001 011001110	{4}
2446	9	2/5	4	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010010 000001111 000001111 000001100 100000010 011100101 011101001 111010001 011001110	{4}
2447	9	2/5	36.1803	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000101 000000101 110000001 001110001 111000001 111111110	{8}
2448	9	2/5	36.1803	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000101 000000101 110000001 001110001 111000001 111111110	{8}
2449	9	2/5	33.4164	{2, 3, 2, 3}	(0, 0)	0	000011011 000001110 000000011 000000011 100000101 110000000 010010001 111100000 101110100	{28}
2450	9	2/5	33.4164	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000001110 000000011 000000011 100000101 110000000 010010001 111100000 101110100	{28}
2451	9	2/5	31.7082	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001011 110000001 001100010 111000010 110101100 111110000	{16}
2452	9	2/5	31.7082	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001011 110000001 001100010 111000010 110101100 111110000	{16}
2453	9	2/5	3.71885	{6, 7, 6, 7}	(0, 0)	0	000011111 000010111 000001111 000000110 110001111 101010111 111111001 111111001 111011110	{4}
2454	9	2/5	3.71885	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000010111 000001111 000000110 110001111 101010111 111111001 111111001 111011110	{4}
2455	9	2/5	3.70075	{0, 1, 4, 3}	(1.5708, 0)	0.96804	000001100 000001010 000000110 000000101 000000011 110000111 101101001 011011001 000111110	{4}
2456	9	2/5	3.70075	{0, 1, 3, 4}	(1.5708, 0)	-2.17355	000001100 000001010 000000110 000000101 000000011 110000111 101101001 011011001 000111110	{4}
2457	9	2/5	3.65836	{1, 2, 1, 2}	(0, 0)	0	000010101 000001111 000001111 000001011 100000011 011100110 111001011 011111101 111110110	{4}
2458	9	2/5	3.65836	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001111 000001111 000001011 100000011 011100110 111001011 011111101 111110110	{4}
2459	9	2/5	3.57044	{2, 3, 2, 3}	(0, 0)	0	000001011 000001001 000000111 000000111 000000101 110000010 001110011 101101101 111110110	{8}
2460	9	2/5	3.57044	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000001001 000000111 000000111 000000101 110000010 001110011 101101101 111110110	{8}
2461	9	2/5	3.12868	{6, 7, 6, 7}	(0, 0)	0	000111111 000011111 000000110 100001111 110001111 110110111 111111001 111111001 110111110	{4}
2462	9	2/5	3.12868	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000111111 000011111 000000110 100001111 110001111 110110111 111111001 111111001 110111110	{4}
2463	9	2/5	3.05573	{1, 4, 2, 3}	(0, 0)	0	000001111 000000101 000000101 000000011 000000011 100000000 111000001 100110001 111110110	{16}
2464	9	2/5	3.05573	{1, 4, 3, 2}	(1.5708, 0)	-3.14159	000001111 000000101 000000101 000000011 000000011 100000000 111000001 100110001 111110110	{16}
2465	9	2/5	28.9443	{7, 8, 7, 8}	(0, 0)	0	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{36}
2466	9	2/5	28.9443	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{36}
2467	9	2/5	25.5279	{2, 3, 2, 3}	(0, 0)	0	000011111 000001100 000000111 000000111 100001001 110010011 111100011 101101101 101111110	{4}
2468	9	2/5	25.5279	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001100 000000111 000000111 100001001 110010011 111100011 101101101 101111110	{4}
2469	9	2/5	24.4721	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
2470	9	2/5	24.4721	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
2471	9	2/5	24.3713	{6, 7, 6, 7}	(0, 0)	0	000011111 000001111 000000111 000000111 100000111 110000111 111111001 111111001 111111110	{4}
2472	9	2/5	24.3713	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000001111 000000111 000000111 100000111 110000111 111111001 111111001 111111110	{4}
2473	9	2/5	22.7639	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000111 000000110 000000101 110000111 001111001 111101001 111011110	{8}
2474	9	2/5	22.7639	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000111 000000110 000000101 110000111 001111001 111101001 111011110	{8}
2475	9	2/5	20	{5, 7, 5, 7}	(0, 0)	0	000001110 000001011 000001011 000000101 000000101 111000001 100110001 111000001 011111110	{36}
2476	9	2/5	20	{5, 7, 7, 5}	(1.5708, 0)	3.14159	000001110 000001011 000001011 000000101 000000101 111000001 100110001 111000001 011111110	{36}
2477	9	2/5	2.91796	{1, 2, 1, 2}	(0, 0)	0	000010101 000001111 000001111 000001010 100000011 011100001 111000010 011110100 111011000	{16}
2478	9	2/5	2.91796	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001111 000001111 000001010 100000011 011100001 111000010 011110100 111011000	{16}
2479	9	2/5	2.71353	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000010 000000001 000000001 100000000 110000010 101000100 110110000	{16}
2480	9	2/5	2.71353	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000010 000000001 000000001 100000000 110000010 101000100 110110000	{16}
2481	9	2/5	2.55279	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000011 100000111 111000101 011011001 111110000 111111100	{16}
2482	9	2/5	2.55279	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000011 100000111 111000101 011011001 111110000 111111100	{16}
2483	9	2/5	2.49001	{4, 5, 7, 6}	(1.5708, 0)	1.40974	000111001 000011111 000011111 100000111 111000101 111000011 011110001 011101001 111111110	{4}
2484	9	2/5	2.49001	{4, 5, 6, 7}	(1.5708, 0)	-1.73185	000111001 000011111 000011111 100000111 111000101 111000011 011110001 011101001 111111110	{4}
2485	9	2/5	2.21115	{6, 7, 6, 7}	(0, 0)	0	000011111 000010110 000001111 000001111 110000000 101100001 111100001 111100001 101101110	{4}
2486	9	2/5	2.21115	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000010110 000001111 000001111 110000000 101100001 111100001 111100001 101101110	{4}
2487	9	2/5	2.05573	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000010 100000001 010000000 001000000 100100000 111010000	{8}
2488	9	2/5	2.05573	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000010 100000001 010000000 001000000 100100000 111010000	{8}
2489	9	2/5	2.04679	{1, 2, 5, 4}	(1.5708, 0)	1.40974	000100101 000010110 000001110 100000101 010000011 001000011 111100011 011011101 100111110	{4}
2490	9	2/5	2.04679	{1, 2, 4, 5}	(1.5708, 0)	-1.73185	000100101 000010110 000001110 100000101 010000011 001000011 111100011 011011101 100111110	{4}
2491	9	2/5	19.9443	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{32}
2492	9	2/5	19.9443	{1, 6, 2, 5}	(1.5708, 0)	-3.14159	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{32}
2493	9	2/5	18.2918	{1, 2, 1, 2}	(0, 0)	0	000010110 000001111 000001111 000000001 100000011 011000101 111001011 111010101 011111110	{4}
2494	9	2/5	18.2918	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010110 000001111 000001111 000000001 100000011 011000101 111001011 111010101 011111110	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2495	9	2/5	17.2361	{2, 3, 2, 3}	(0, 0)	0	000001011 000001010 000000101 000000101 000000001 110000011 001100000 110001001 101111010	{48}
2496	9	2/5	17.2361	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000001010 000000101 000000101 000000001 110000011 001100000 110001001 101111010	{48}
2497	9	2/5	16.5836	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001011 110001001 001110010 111000010 110101100 111110000	{8}
2498	9	2/5	16.5836	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001011 110001001 001110010 111000010 110101100 111110000	{8}
2499	9	2/5	15.5279	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001111 000001011 110000000 001100011 111000001 111101001 111101110	{8}
2500	9	2/5	15.5279	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001111 000001011 110000000 001100011 111000001 111101001 111101110	{8}
2501	9	2/5	15.4721	{0, 4, 3, 1}	(0, 0)	0	000101111 000011111 000000011 100001111 010001111 110110010 110110001 111111000 111110100	{16}
2502	9	2/5	15.4721	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	000101111 000011111 000000011 100001111 010001111 110110010 110110001 111111000 111110100	{16}
2503	9	2/5	14.2082	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000011 000000001 000000001 100000000 110000010 101000100 111110000	{16}
2504	9	2/5	14.2082	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000011 000000001 000000001 100000000 110000010 101000100 111110000	{16}
2505	9	2/5	13.8197	{1, 2, 1, 2}	(0, 0)	0	000011011 000000111 000000111 000000001 100001001 100010001 011000010 111000100 111111000	{8}
2506	9	2/5	13.8197	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000000111 000000111 000000001 100001001 100010001 011000010 111000100 111111000	{8}
2507	9	2/5	13.7812	{1, 2, 1, 2}	(0, 0)	0	000010100 000001111 000001111 000001000 100000010 011100001 111000001 011010001 011001110	{8}
2508	9	2/5	13.7812	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010100 000001111 000001111 000001000 100000010 011100001 111000001 011010001 011001110	{8}
2509	9	2/5	12.7639	{1, 2, 1, 2}	(0, 0)	0	000001011 000000101 000000101 000000011 000000011 100000010 011000001 100111001 111110110	{60}
2510	9	2/5	12.7639	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000101 000000101 000000011 000000011 100000010 011000001 100111001 111110110	{60}
2511	9	2/5	12.6631	{6, 7, 6, 7}	(0, 0)	0	000111111 000010111 000001111 100011111 110101111 101110111 111111001 111111001 111111110	{4}
2512	9	2/5	12.6631	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000111111 000010111 000001111 100011111 110101111 101110111 111111001 111111001 111111110	{4}
2513	9	2/5	12.1115	{2, 3, 2, 3}	(0, 0)	0	000011101 000001100 000000111 000000111 100000011 110000010 111100001 001111001 101110110	{12}
2514	9	2/5	12.1115	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000001100 000000111 000000111 100000011 110000010 111100001 001111001 101110110	{12}
2515	9	2/5	11.8541	{0, 4, 5, 1}	(0, 0)	0	000010010 000001010 000000110 000000011 100000001 010000001 001000001 111100001 000111110	{32}
2516	9	2/5	11.8541	{0, 4, 1, 5}	(1.5708, 0)	3.14159	000010010 000001010 000000110 000000011 100000001 010000001 001000001 111100001 000111110	{32}
2517	9	2/5	11.0557	{3, 4, 3, 4}	(0, 0)	0	000001011 000001010 000000110 000000101 000000101 110000010 001110001 111001001 100110110	{28}
2518	9	2/5	11.0557	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001011 000001010 000000110 000000101 000000101 110000010 001110001 111001001 100110110	{28}
2519	9	2/5	1.86951	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001100 100000011 011100111 111101001 111011000 111011100	{4}
2520	9	2/5	1.86951	{1, 2, 2, 1}	(1.5708, 0)	3.14159	000010111 000001111 000001111 000001100 100000011 011100111 111101001 111011000 111011100	{4}
2521	9	2/5	1.86223	{6, 7, 6, 7}	(0, 0)	0	000101111 000010111 000001110 100001111 010000001 101100001 111100001 111100001 110111110	{4}
2522	9	2/5	1.86223	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000101111 000010111 000001110 100001111 010000001 101100001 111100001 111100001 110111110	{4}
2523	9	2/5	1.84262	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000110 000000101 000000011 110000111 111101011 111011101 110111110	{8}
2524	9	2/5	1.84262	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000110 000000101 000000011 110000111 111101011 111011101 110111110	{8}
2525	9	2/5	1.61301	{6, 7, 6, 7}	(0, 0)	0	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{8}
2526	9	2/5	1.61301	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{8}
2527	9	2/5	1.52786	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000001101 100000010 011100101 011101000 111010000 111101000	{4}
2528	9	2/5	1.52786	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000001101 100000010 011100101 011101000 111010000 111101000	{4}
2529	9	2/5	1.31672	{2, 3, 2, 3}	(0, 0)	0	000001111 000001101 000000011 000000011 000000010 110000101 110001001 101110000 111101100	{16}
2530	9	2/5	1.31672	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000001101 000000011 000000011 000000010 110000101 110001001 101110000 111101100	{16}
2531	9	2/5	1.11456	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001001 100000111 011100001 111010011 111010101 111111110	{4}
2532	9	2/5	1.11456	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000001001 100000111 011100001 111010011 111010101 111111110	{4}
2533	9	2/5	1.03648	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000010 000000001 000000001 100000001 110000010 101000100 110111000	{16}
2534	9	2/5	1.03648	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000010 000000001 000000001 100000001 110000010 101000100 110111000	{16}
2535	9	2/5	0.937812	{2, 3, 2, 3}	(0, 0)	0	000011111 000011111 000001111 000001111 110000011 111100100 111101000 111110001 111110010	{4}
2536	9	2/5	0.937812	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001111 110000011 111100100 111101000 111110001 111110010	{4}
2537	9	2/5	0.844582	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000001010 100000111 111100011 111010001 011111001 111011110	{12}
2538	9	2/5	0.844582	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000001010 100000111 111100011 111010001 011111001 111011110	{12}
2539	9	2/5	0.791796	{3, 4, 3, 4}	(0, 0)	0	000001101 000000100 000000011 000000010 000000010 100000000 110000001 001110001 101000110	{16}
2540	9	2/5	0.791796	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001101 000000100 000000011 000000010 000000010 100000000 110000001 001110001 101000110	{16}
2541	9	2/5	0.767997	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000001101 100000111 111100001 011110001 111010001 111111110	{4}
2542	9	2/5	0.767997	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000001101 100000111 111100001 011110001 111010001 111111110	{4}
2543	9	2/5	0.71131	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000001010 100000111 111100011 111010000 011111001 111011010	{4}
2544	9	2/5	0.71131	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000001010 100000111 111100011 111010000 011111001 111011010	{4}
2545	9	1/3	9	{6, 1, 3, 4}	(0, 0)	2.0944	000001011 000001010 000000110 000000100 000000011 110000101 001101001 111010000 100011100	{52}
2546	9	1/3	9	{4, 8, 7, 5}	(0, 0)	0	000000111 000000111 000000011 000000011 000000010 000000001 110000000 111110000 111101000	{32}
2547	9	1/3	9	{1, 6, 4, 3}	(0, 0)	-2.0944	000001011 000001010 000000110 000000100 000000011 110000101 001101001 111010000 100011100	{52}
2548	9	1/3	9	{1, 6, 3, 4}	(1.5708, 1.0472)	-3.14159	000001011 000001010 000000110 000000100 000000011 110000101 001101001 111010000 100011100	{52}
2549	9	1/3	9	{4, 8, 5, 7}	(1.5708, 0)	3.14159	000000111 000000111 000000011 000000011 000000010 000000001 110000000 111110000 111101000	{32}
2550	9	1/3	9	{6, 1, 4, 3}	(1.5708, -1.0472)	3.14159	000001011 000001010 000000110 000000100 000000011 110000101 001101001 111010000 100011100	{52}
2551	9	1/3	8	{1, 8, 7, 8}	(0, 0)	-3.14159	000010100 000001111 000001011 000000001 100000011 011000110 110001001 011011000 011110100	{16}
2552	9	1/3	8	{8, 1, 7, 8}	(1.5708, -1.5708)	3.14159	000010100 000001111 000001011 000000001 100000011 011000110 110001001 011011000 011110100	{16}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2553	9	1/3	7.75	{2, 3, 2, 3}	(0, 0)	0	000010110 000001110 000000011 000000011 100000101 010000001 110010001 111100000 001111100	{20}
2554	9	1/3	7.75	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010110 000001110 000000011 000000011 100000101 010000001 110010001 111100000 001111100	{20}
2555	9	1/3	7.5	{0, 8, 3, 7}	(0, 0)	2.0944	000100111 000010101 000001011 100000100 010000010 001000011 110100001 101011000 111001100	{4}
2556	9	1/3	7.5	{6, 7, 6, 7}	(0, 0)	0	000001111 000000111 000000111 000000111 000000111 000000111 100000001 111110001 111110001 111111110	{8}
2557	9	1/3	7.5	{8, 0, 7, 3}	(0, 0)	-2.0944	000100111 000010101 000001011 100000100 010000010 001000011 110100001 101011000 111001100	{4}
2558	9	1/3	7.5	{8, 0, 3, 7}	(1.5708, 1.0472)	-3.14159	000100111 000010101 000001011 100000100 010000010 001000011 110100001 101011000 111001100	{4}
2559	9	1/3	7.5	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000001111 000000111 000000111 000000111 000000111 000000111 100000001 111110001 111110001 111111110	{8}
2560	9	1/3	7.5	{0, 8, 7, 3}	(1.5708, -1.0472)	3.14159	000100111 000010101 000001011 100000100 010000010 001000011 110100001 101011000 111001100	{4}
2561	9	1/3	6.5	{4, 0, 1, 0}	(0, 0)	2.0944	000011110 000001110 000001011 000000101 100000001 111000011 110100000 111001001 001111010	{10}
2562	9	1/3	6.5	{3, 2, 3, 5}	(0, 0)	1.0472	000011011 000010100 000001111 000000111 110000011 101000000 011100000 101110001 101110010	{6}
2563	9	1/3	6.5	{0, 4, 0, 1}	(0, 0)	-2.0944	000011110 000001110 000001011 000000101 100000001 111000011 110100000 111001001 001111010	{10}
2564	9	1/3	6.5	{2, 3, 5, 3}	(0, 0)	-1.0472	000011011 000010100 000001111 000000111 110000011 101000000 011100000 101110001 101110010	{6}
2565	9	1/3	6.5	{0, 4, 1, 0}	(1.5708, 1.0472)	-3.14159	000011110 000001110 000001011 000000101 100000001 111000011 110100000 111001001 001111010	{10}
2566	9	1/3	6.5	{2, 3, 3, 5}	(1.5708, 0.523599)	-3.14159	000011011 000010100 000001111 000000111 110000011 101000000 011100000 101110001 101110010	{6}
2567	9	1/3	6.5	{4, 0, 0, 1}	(1.5708, -1.0472)	3.14159	000011110 000001110 000001011 000000101 100000001 111000011 110100000 111001001 001111010	{10}
2568	9	1/3	6.5	{3, 2, 5, 3}	(1.5708, -0.523599)	3.14159	000011011 000010100 000001111 000000111 110000011 101000000 011100000 101110001 101110010	{6}
2569	9	1/3	6.44444	{2, 3, 2, 3}	(0, 0)	0	000011101 000001110 000000011 000000011 100000011 110000111 110001000 011111001 101111010	{8}
2570	9	1/3	6.44444	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000001110 000000011 000000011 100000011 110000111 110001000 011111001 101111010	{8}
2571	9	1/3	6.25	{0, 5, 4, 1}	(0, 0)	0	000010011 000001011 000000110 000000001 100000011 010000011 001000001 111011000 110111100	{16}
2572	9	1/3	6.25	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	000010011 000001011 000000110 000000001 100000011 010000011 001000001 111011000 110111100	{16}
2573	9	1/3	6	{2, 0, 2, 1}	(0, 0)	2.0944	000000101 000000101 000000011 000000010 000000010 000000001 110000010 001110101 111001010	{173}
2574	9	1/3	6	{5, 0, 1, 2}	(0, 0)	1.0472	000001010 000000111 000000101 000000011 000000001 100000001 011000000 110100000 011111000	{58}
2575	9	1/3	6	{5, 1, 3, 1}	(0, 0)	-3.14159	000001010 000000101 000000100 000000011 000000001 100000001 011000010 100100101 010111010	{53}
2576	9	1/3	6	{0, 2, 1, 2}	(0, 0)	-2.0944	000000101 000000101 000000011 000000010 000000010 000000001 110000010 001110101 111001010	{173}
2577	9	1/3	6	{0, 5, 2, 1}	(0, 0)	-1.0472	000001010 000000111 000000101 000000011 000000001 100000001 011000000 110100000 011111000	{58}
2578	9	1/3	6	{0, 2, 2, 1}	(1.5708, 1.0472)	-3.14159	000000101 000000101 000000011 000000010 000000010 000000001 110000010 001110101 111001010	{173}
2579	9	1/3	6	{0, 5, 1, 2}	(1.5708, 0.523599)	-3.14159	000001010 000000111 000000101 000000011 000000001 100000001 011000000 110100000 011111000	{58}
2580	9	1/3	6	{1, 5, 3, 1}	(1.5708, -1.5708)	3.14159	000001010 000000101 000000100 000000011 000000001 100000001 011000010 100100101 010111010	{53}
2581	9	1/3	6	{2, 0, 1, 2}	(1.5708, -1.0472)	3.14159	000000101 000000101 000000011 000000010 000000010 000000001 110000010 001110101 111001010	{173}
2582	9	1/3	6	{5, 0, 2, 1}	(1.5708, -0.523599)	3.14159	000001010 000000111 000000101 000000011 000000001 100000001 011000000 110100000 011111000	{58}
2583	9	1/3	58	{1, 2, 1, 2}	(0, 0)	0	000010101 000001010 000001010 000000100 100000011 011000111 100101001 011011001 100011110	{8}
2584	9	1/3	58	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010101 000001010 000001010 000000100 100000011 011000111 100101001 011011001 100011110	{8}
2585	9	1/3	52	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001000 110000010 001100011 111000001 110011001 111001110	{4}
2586	9	1/3	52	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001000 110000010 001100011 111000001 110011001 111001110	{4}
2587	9	1/3	5.8	{0, 4, 3, 1}	(0, 0)	0	000100111 000010111 000001101 100000111 010000111 001000010 111110001 110111000 111110100	{8}
2588	9	1/3	5.8	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	000100111 000010111 000001101 100000111 010000111 001000010 111110001 110111000 111110100	{8}
2589	9	1/3	5.77778	{2, 3, 2, 3}	(0, 0)	0	000011111 000010011 000001111 000001111 110000101 101100110 101111000 111101001 111110010	{4}
2590	9	1/3	5.77778	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000010011 000001111 000001111 110000101 101100110 101111000 111101001 111110010	{4}
2591	9	1/3	5.75	{7, 8, 7, 8}	(0, 0)	0	000001111 000001000 000000111 000000011 000000011 110000011 101000000 101111000 101111000	{32}
2592	9	1/3	5.75	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001111 000001000 000000111 000000011 000000011 110000011 101000000 101111000 101111000	{32}
2593	9	1/3	5.5	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000101 000000011 000000011 100000000 111000001 100110000 111110100	{52}
2594	9	1/3	5.5	{0, 6, 3, 6}	(0, 0)	-3.14159	000011011 000001111 000001100 000000011 100000110 111000001 011010001 110110001 110101110	{8}
2595	9	1/3	5.5	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000101 000000011 000000011 100000000 111000001 100110000 111110100	{52}
2596	9	1/3	5.5	{6, 0, 3, 6}	(1.5708, -1.5708)	3.14159	000011011 000001111 000001100 000000011 100000110 111000001 011010001 110110001 110101110	{8}
2597	9	1/3	5.25	{4, 7, 8, 6}	(1.5708, -2.0944)	-1.0472	000011111 000010011 000001000 000000111 110001110 101010111 100111001 110111001 110101110	{2}
2598	9	1/3	5.25	{7, 4, 6, 8}	(1.5708, 2.0944)	-1.0472	000011111 000010011 000001000 000000111 110001110 101010111 100111001 110111001 110101110	{2}
2599	9	1/3	5.25	{7, 4, 8, 6}	(1.10715, 3.14159)	2.63623	000011111 000010011 000001000 000000111 110001110 101010111 100111001 110111001 110101110	{2}
2600	9	1/3	5.25	{4, 7, 6, 8}	(1.10715, 0)	-2.63623	000011111 000010011 000001000 000000111 110001110 101010111 100111001 110111001 110101110	{2}
2601	9	1/3	42	{1, 2, 1, 2}	(0, 0)	0	000010011 000001101 000001101 000000100 100000010 011000011 011100010 100011101 111001010	{28}
2602	9	1/3	42	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001101 000001101 000000100 100000010 011000011 011100010 100011101 111001010	{28}
2603	9	1/3	40	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001011 110000000 001100011 111000000 110101000 111101000	{8}
2604	9	1/3	40	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001011 110000000 001100011 111000000 110101000 111101000	{8}
2605	9	1/3	4.75	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001101 000001010 110000100 001100011 111010001 110101001 111001110	{4}
2606	9	1/3	4.75	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001101 000001010 110000100 001100011 111010001 110101001 111001110	{4}
2607	9	1/3	4.71429	{0, 1, 6, 3}	(1.5708, 0)	1.42745	000110111 000101111 000011001 110001011 101000111 011100011 110010011 110111101 111111110	{4}
2608	9	1/3	4.71429	{0, 1, 3, 6}	(1.5708, 0)	-1.71414	000110111 000101111 000011001 110001011 101000111 011100011 110010011 110111101 111111110	{4}
2609	9	1/3	4.66667	{1, 2, 1, 2}	(0, 0)	0	000010011 000001101 000001101 000000101 100000010 011000010 011100011 100011101 111100110	{16}
2610	9	1/3	4.66667	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001101 000001101 000000101 100000010 011000010 011100011 100011101 111100110	{16}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2611	9	1/3	4.5	{2, 0, 2, 1}	(0, 0)	2.0944	000001110 000001110 000000111 000000001 000000001 110000001 111000011 111000101 001111110	{73}
2612	9	1/3	4.5	{1, 7, 6, 7}	(0, 0)	1.0472	000010111 000001010 000000101 000000001 100000111 010000001 101010000 110010001 101111010	{22}
2613	9	1/3	4.5	{2, 8, 4, 8}	(0, 0)	-3.14159	000010111 000001011 000000101 000000001 100000001 010000011 101000011 110001100 111111100	{83}
2614	9	1/3	4.5	{0, 2, 1, 2}	(0, 0)	-2.0944	000001110 000001110 000000111 000000001 000000001 110000001 111000011 111000101 001111110	{73}
2615	9	1/3	4.5	{7, 1, 7, 6}	(0, 0)	-1.0472	000010111 000001010 000000101 000000001 100000111 010000001 101010000 110010001 101111010	{22}
2616	9	1/3	4.5	{0, 2, 2, 1}	(1.5708, 1.0472)	-3.14159	000001110 000001110 000000111 000000001 000000001 110000001 111000011 111000101 001111110	{73}
2617	9	1/3	4.5	{7, 1, 6, 7}	(1.5708, 0.523599)	-3.14159	000010111 000001010 000000101 000000001 100000111 010000001 101010000 110010001 101111010	{22}
2618	9	1/3	4.5	{8, 2, 4, 8}	(1.5708, -1.5708)	3.14159	000010111 000001011 000000101 000000001 100000001 010000011 101000011 110001100 111111100	{83}
2619	9	1/3	4.5	{2, 0, 1, 2}	(1.5708, -1.0472)	3.14159	000001110 000001110 000000111 000000001 000000001 110000001 111000011 111000101 001111110	{73}
2620	9	1/3	4.5	{1, 7, 7, 6}	(1.5708, -0.523599)	3.14159	000010111 000001010 000000101 000000001 100000111 010000001 101010000 110010001 101111010	{22}
2621	9	1/3	4.42857	{2, 3, 5, 6}	(1.5708, 3.14159)	-1.42745	000011101 000001101 000001010 000000110 100000011 111000001 110100001 001110000 110011100	{4}
2622	9	1/3	4.42857	{2, 3, 6, 5}	(1.5708, -3.14159)	1.71414	000011101 000001101 000001010 000000110 100000011 111000001 110100001 001110000 110011100	{4}
2623	9	1/3	4.4	{7, 8, 7, 8}	(0, 0)	0	000010111 000001100 000000011 000000011 100000111 010000011 110010011 101111100 101111100	{4}
2624	9	1/3	4.4	{7, 8, 8, 7}	(1.5708, 0)	-3.14159	000010111 000001100 000000011 000000011 100000111 010000011 110010011 101111100 101111100	{4}
2625	9	1/3	4.125	{2, 3, 2, 3}	(0, 0)	0	000011111 000001111 000000011 000000011 100000010 110000101 110001001 111110001 111101110	{24}
2626	9	1/3	4.125	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001111 000000011 000000011 100000010 110000101 110001001 111110001 111101110	{24}
2627	9	1/3	4	{7, 0, 5, 0}	(0, 0)	2.47465	000111111 000011111 000000111 100011011 110100111 110100101 111011000 111110001 111111010	{2}
2628	9	1/3	4	{0, 3, 6, 2}	(0, 0)	0.666946	000011101 000010011 000001111 000001111 110000111 101100011 101110001 011111000 111111100	{4}
2629	9	1/3	4	{0, 7, 0, 5}	(0, 0)	-2.47465	000111111 000011111 000000111 100011011 110100111 110100101 111011000 111110001 111111010	{2}
2630	9	1/3	4	{3, 0, 2, 6}	(0, 0)	-0.666946	000011101 000010011 000001111 000001111 110000111 101100011 101110001 011111000 111111100	{4}
2631	9	1/3	4	{5, 2, 4, 6}	(0.857072, -1.5708)	-1.44547	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{2}
2632	9	1/3	4	{2, 5, 6, 4}	(0.857072, -1.5708)	1.44547	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{2}
2633	9	1/3	4	{6, 4, 2, 5}	(0.857072, 1.5708)	1.44547	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{2}
2634	9	1/3	4	{3, 5, 1, 2}	(0.857072, -2.61799)	-1.44547	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2635	9	1/3	4	{1, 2, 3, 5}	(0.857072, 2.61799)	-1.44547	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2636	9	1/3	4	{2, 1, 5, 3}	(0.857072, 0.523599)	1.44547	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2637	9	1/3	4	{4, 6, 5, 2}	(0.857072, 1.5708)	-1.44547	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{2}
2638	9	1/3	4	{5, 3, 2, 1}	(0.857072, -0.523599)	1.44547	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2639	9	1/3	4	{2, 3, 6, 5}	(1.5708, 3.14159)	-1.42745	000010011 000001110 000001001 000000101 100000010 011000001 010100001 110010000 101101100	{4}
2640	9	1/3	4	{2, 3, 5, 6}	(1.5708, -3.14159)	1.71414	000010011 000001110 000001001 000000101 100000010 011000001 010100001 110010000 101101100	{4}
2641	9	1/3	4	{5, 3, 1, 2}	(1.28976, 2.61799)	2.24593	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2642	9	1/3	4	{4, 6, 2, 5}	(1.0472, -2.61799)	3.14159	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{2}
2643	9	1/3	4	{1, 2, 5, 3}	(1.28976, -2.61799)	2.24593	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2644	9	1/3	4	{2, 1, 3, 5}	(1.28976, -0.523599)	-2.24593	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2645	9	1/3	4	{3, 5, 2, 1}	(1.28976, 0.523599)	-2.24593	000001110 000001001 000000111 000000110 000000001 110000011 101100001 101101001 011011110	{3}
2646	9	1/3	4	{6, 4, 5, 2}	(1.0472, -0.523599)	-3.14159	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{1}
2647	9	1/3	4	{2, 5, 4, 6}	(1.0472, 2.61799)	3.14159	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{2}
2648	9	1/3	4	{5, 2, 6, 4}	(1.0472, 0.523599)	-3.14159	000010101 000001111 000001010 000000001 100000011 011000101 110001010 011010100 110111000	{1}
2649	9	1/3	4	{0, 7, 5, 0}	(1.5708, 1.23732)	-3.14159	000111111 000011111 000000111 100011011 110100111 110100101 111011000 111110001 111111010	{2}
2650	9	1/3	4	{3, 0, 6, 2}	(1.5708, 0.333473)	-3.14159	000011101 000010011 000001111 000001111 110000111 101100011 101110001 011111000 111111100	{4}
2651	9	1/3	4	{7, 0, 0, 5}	(1.5708, -1.23732)	3.14159	000111111 000011111 000000111 100011011 110100111 110100101 111011000 111110001 111111010	{2}
2652	9	1/3	4	{0, 3, 2, 6}	(1.5708, -0.333473)	3.14159	000011101 000010011 000001111 000001111 110000111 101100011 101110001 011111000 111111100	{4}
2653	9	1/3	38.5	{4, 5, 4, 5}	(0, 0)	0	000100111 000011101 000011010 100000111 011000011 011000011 110100001 101111001 110111110	{4}
2654	9	1/3	38.5	{4, 5, 5, 4}	(1.5708, 0)	3.14159	000100111 000011101 000011010 100000111 011000011 011000011 110100001 101111001 110111110	{4}
2655	9	1/3	36	{3, 4, 3, 4}	(0, 0)	0	000001111 000001100 000001011 000000111 000000111 111000010 110110001 101111001 101110110	{12}
2656	9	1/3	36	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001100 000001011 000000111 000000111 111000010 110110001 101111001 101110110	{12}
2657	9	1/3	30	{0, 1, 0, 1}	(0, 0)	0	000010111 000010111 000001100 000001011 110000000 001100011 111000000 110101000 110101000	{4}
2658	9	1/3	30	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000010111 000010111 000001100 000001011 110000000 001100011 111000000 110101000 110101000	{4}
2659	9	1/3	3.83333	{7, 8, 7, 8}	(0, 0)	0	000001010 000001001 000000111 000000111 000000011 110000111 001101000 101111001 011111010	{36}
2660	9	1/3	3.83333	{7, 8, 8, 7}	(1.5708, 0)	-3.14159	000001010 000001001 000000111 000000111 000000011 110000111 001101000 101111001 011111010	{36}
2661	9	1/3	3.75	{0, 1, 0, 1}	(0, 0)	0	000000111 000000111 000000100 000000011 000000001 011000001 110111001 110100000 110111100	{68}
2662	9	1/3	3.75	{0, 4, 6, 2}	(1.5708, 0)	-1.0472	000011010 000010111 000001101 000000011 110000010 101000011 011000001 110111001 011101110	{8}
2663	9	1/3	3.75	{5, 8, 6, 7}	(1.5708, -2.0944)	-1.0472	000111111 000001101 000000111 100011111 100101111 110110011 111110011 101111100 111111100	{2}
2664	9	1/3	3.75	{0, 5, 3, 2}	(1.5708, 0)	1.0472	000100110 000010111 000001011 100000100 010000000 001000001 110100001 111000000 011001100	{4}
2665	9	1/3	3.75	{8, 5, 7, 6}	(1.5708, 2.0944)	-1.0472	000111111 000001101 000000111 100011111 100101111 110110011 111110011 101111100 111111100	{2}
2666	9	1/3	3.75	{8, 5, 6, 7}	(1.10715, 3.14159)	2.63623	000111111 000001101 000000111 100011111 100101111 110110011 111110011 101111100 111111100	{2}
2667	9	1/3	3.75	{0, 5, 2, 3}	(1.5708, 0)	-2.0944	000100110 000010111 000001011 100000100 010000000 001000001 110100001 111000000 011001100	{4}
2668	9	1/3	3.75	{5, 8, 7, 6}	(1.10715, 0)	-2.63623	000111111 000001101 000000111 100011111 100101111 110110011 111110011 101111100 111111100	{2}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2669	9	1/3	3.75	{0, 4, 2, 6}	(1.5708, -3.14159)	-2.0944	000011010 000010111 000001101 000000011 110000010 101000011 011000001 110111001 011101110	{8}
2670	9	1/3	3.75	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000000111 000000111 000000100 000000011 000000001 000000001 111000001 110100000 110111100	{68}
2671	9	1/3	3.625	{2, 3, 2, 3}	(0, 0)	0	000011101 000001110 000000011 000000011 100000011 110000100 110001000 011110001 101110010	{28}
2672	9	1/3	3.625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000001110 000000011 000000011 100000011 110000100 110001000 011110001 101110010	{28}
2673	9	1/3	3.6	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000011 000000001 000000001 110000101 110001001 111000000 111111100	{16}
2674	9	1/3	3.6	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000011 000000001 000000001 110000101 110001001 111000000 111111100	{16}
2675	9	1/3	3.5	{1, 7, 2, 3}	(0, 0)	2.0944	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{92}
2676	9	1/3	3.5	{8, 1, 3, 2}	(0, 0)	1.0472	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{36}
2677	9	1/3	3.5	{7, 1, 3, 2}	(0, 0)	-2.0944	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{92}
2678	9	1/3	3.5	{1, 8, 2, 3}	(0, 0)	-1.0472	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{36}
2679	9	1/3	3.5	{8, 5, 6, 7}	(1.5708, 2.0944)	1.0472	000001010 000001001 000000111 000000100 000000011 110000010 001100001 101011000 011010100	{4}
2680	9	1/3	3.5	{5, 2, 6, 4}	(0.523599, -2.61799)	-3.14159	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2681	9	1/3	3.5	{4, 6, 5, 2}	(0.857072, -1.5708)	-1.44547	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2682	9	1/3	3.5	{5, 8, 7, 6}	(1.5708, -2.0944)	1.0472	000001010 000001001 000000111 000000100 000000011 110000010 001100001 101011000 011010100	{4}
2683	9	1/3	3.5	{6, 4, 2, 5}	(0.857072, -1.5708)	1.44547	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2684	9	1/3	3.5	{2, 5, 4, 6}	(0.523599, -0.523599)	3.14159	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2685	9	1/3	3.5	{6, 4, 5, 2}	(0.523599, 2.61799)	-3.14159	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2686	9	1/3	3.5	{2, 5, 6, 4}	(0.857072, 1.5708)	1.44547	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2687	9	1/3	3.5	{5, 2, 4, 6}	(0.857072, 1.5708)	-1.44547	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2688	9	1/3	3.5	{4, 6, 2, 5}	(0.523599, 0.523599)	3.14159	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2689	9	1/3	3.5	{5, 2, 6, 4}	(1.0472, -2.61799)	3.14159	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2690	9	1/3	3.5	{4, 6, 5, 2}	(1.28976, -1.5708)	-2.24593	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2691	9	1/3	3.5	{6, 4, 2, 5}	(1.28976, -1.5708)	2.24593	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2692	9	1/3	3.5	{2, 5, 4, 6}	(1.0472, -0.523599)	3.14159	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2693	9	1/3	3.5	{6, 4, 5, 2}	(1.0472, 2.61799)	3.14159	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2694	9	1/3	3.5	{2, 5, 6, 4}	(1.28976, 1.5708)	2.24593	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2695	9	1/3	3.5	{5, 8, 6, 7}	(1.10715, -3.14159)	-2.63623	000001010 000001001 000000111 000000100 000000011 110000010 001100001 101011000 011010100	{4}
2696	9	1/3	3.5	{5, 2, 4, 6}	(1.28976, 1.5708)	-2.24593	000010111 000001111 000001010 000000011 100000001 011000101 110001000 111100000 110111000	{1}
2697	9	1/3	3.5	{4, 6, 2, 5}	(1.0472, 0.523599)	3.14159	000010110 000001101 000001011 000000011 100000001 011000110 110001011 101101101 011110110	{3}
2698	9	1/3	3.5	{8, 5, 7, 6}	(1.10715, 0)	2.63623	000001010 000001001 000000111 000000100 000000011 110000010 001100001 101011000 011010100	{4}
2699	9	1/3	3.5	{7, 1, 2, 3}	(1.5708, 1.0472)	-3.14159	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{92}
2700	9	1/3	3.5	{1, 8, 3, 2}	(1.5708, 0.523599)	-3.14159	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{36}
2701	9	1/3	3.5	{1, 7, 3, 2}	(1.5708, -1.0472)	3.14159	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{92}
2702	9	1/3	3.5	{8, 1, 2, 3}	(1.5708, -0.523599)	3.14159	000001011 000000111 000000100 000000001 000000001 100000011 011000011 110001100 110111100	{36}
2703	9	1/3	3.44444	{2, 8, 7, 6}	(0, 0)	2.0944	000101110 000011101 000000111 100001011 010000111 110100001 111010000 101110001 011111010	{4}
2704	9	1/3	3.44444	{8, 2, 6, 7}	(0, 0)	-2.0944	000101110 000011101 000000111 100001011 010000111 110100001 111010000 101110001 011111010	{4}
2705	9	1/3	3.44444	{8, 2, 7, 6}	(1.5708, 1.0472)	-3.14159	000101110 000011101 000000111 100001011 010000111 110100001 111010000 101110001 011111010	{4}
2706	9	1/3	3.44444	{2, 8, 6, 7}	(1.5708, -1.0472)	3.14159	000101110 000011101 000000111 100001011 010000111 110100001 111010000 101110001 011111010	{4}
2707	9	1/3	3.42857	{0, 3, 4, 1}	(1.5708, 0)	-1.42745	000101010 000011001 000001111 100000110 010000101 111000011 001110011 101101100 011011100	{4}
2708	9	1/3	3.42857	{0, 3, 1, 4}	(1.5708, 0)	1.71414	000101010 000011001 000001111 100000110 010000101 111000011 001110011 101101100 011011100	{4}
2709	9	1/3	3.33333	{2, 3, 2, 3}	(0, 0)	0	000011101 000010011 000001111 000001111 110000011 101100000 101100000 011110000 111110000	{12}
2710	9	1/3	3.33333	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000010011 000001111 000001111 110000011 101100000 101100000 011110000 111110000	{12}
2711	9	1/3	3.25	{2, 6, 7, 3}	(0, 0)	0	000010110 000001111 000000101 000000011 100000001 010000000 111000000 110100000 011110000	{76}
2712	9	1/3	3.25	{1, 2, 5, 4}	(0.588003, -3.14159)	2.24593	000011110 000001110 000001111 000000011 100000001 011000001 111000001 111110001 001111110	{2}
2713	9	1/3	3.25	{2, 3, 6, 5}	(1.5708, 0)	-1.0472	000011100 000011100 000001010 000000110 110000011 111000001 110100001 001110001 000011110	{24}
2714	9	1/3	3.25	{4, 5, 0, 6}	(0.588003, -2.0944)	2.24593	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2715	9	1/3	3.25	{7, 5, 3, 4}	(0.523599, -1.5708)	-3.14159	000101110 000010101 000001110 100000011 010000010 101000001 111000001 101110001 010101110	{1}
2716	9	1/3	3.25	{5, 4, 6, 0}	(0.588003, -1.0472)	-2.24593	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2717	9	1/3	3.25	{2, 1, 4, 5}	(0.588003, 0)	-2.24593	000011110 000001110 000000111 000000011 100000011 110000001 111000001 111110001 001111110	{2}
2718	9	1/3	3.25	{0, 6, 4, 5}	(0.588003, 2.0944)	2.24593	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2719	9	1/3	3.25	{3, 4, 7, 5}	(0.523599, 1.5708)	-3.14159	000101110 000010101 000001110 100000011 010000010 101000001 111000001 101110001 010101110	{1}
2720	9	1/3	3.25	{0, 5, 3, 2}	(1.5708, 0)	1.0472	000100110 000010111 000001010 100000101 010000000 001000001 110100000 111000001 010101010	{4}
2721	9	1/3	3.25	{6, 0, 5, 4}	(0.588003, 1.0472)	-2.24593	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2722	9	1/3	3.25	{6, 0, 4, 5}	(1.10715, -2.0944)	-2.63623	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2723	9	1/3	3.25	{0, 5, 2, 3}	(1.5708, 0)	-2.0944	000100110 000010111 000001010 100000101 010000000 001000001 110100000 111000001 010101010	{4}
2724	9	1/3	3.25	{3, 4, 5, 7}	(1.0472, -1.5708)	-3.14159	000101110 000010101 000001110 100000011 010000010 101000001 111000001 101110001 010101110	{1}
2725	9	1/3	3.25	{0, 6, 5, 4}	(1.10715, -1.0472)	2.63623	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2726	9	1/3	3.25	{2, 1, 5, 4}	(1.5708, -2.0944)	-2.0944	000011110 000001110 000000111 000000011 100000011 110000001 111000001 111110001 001111110	{2}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2727	9	1/3	3.25	{4, 5, 6, 0}	(1.10715, 2.0944)	-2.63623	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2728	9	1/3	3.25	{5, 7, 3, 4}	(1.0472, 1.5708)	-3.14159	000101110 000010101 000001110 100000011 010000010 101000001 111000001 101110001 010101110	{1}
2729	9	1/3	3.25	{5, 4, 0, 6}	(1.10715, 1.0472)	2.63623	000011100 000001111 000000111 000000010 100000011 110000001 111000001 011110000 011011100	{1}
2730	9	1/3	3.25	{2, 3, 5, 6}	(1.5708, -3.14159)	-2.0944	000011100 000011100 000001010 000000110 110000011 111000001 110100001 001110001 000011110	{24}
2731	9	1/3	3.25	{1, 2, 4, 5}	(1.5708, 2.0944)	-2.0944	000011110 000001110 000000111 000000011 100000011 110000001 111000001 111110001 001111110	{2}
2732	9	1/3	3.25	{2, 6, 3, 7}	(1.5708, 0)	-3.14159	000010110 000001111 000000101 000000011 100000001 010000000 111000000 110100000 011110000	{76}
2733	9	1/3	3.11111	{2, 3, 2, 3}	(0, 0)	0	000011011 000001111 000000111 000000111 100000010 110000000 011100001 111110000 111100100	{64}
2734	9	1/3	3.11111	{0, 1, 2, 1}	(0, 0)	-3.14159	000101110 000010110 000001111 100000001 010000011 101000100 111001001 111010001 001110110	{9}
2735	9	1/3	3.11111	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000001111 000000111 000000111 100000010 110000000 011100001 111110000 111100100	{64}
2736	9	1/3	3.11111	{1, 0, 2, 1}	(1.5708, -1.5708)	3.14159	000101110 000010110 000001111 100000001 010000011 101000100 111001001 111010001 001110110	{9}
2737	9	1/3	3	{2, 0, 1, 0}	(0, 0)	-3.14159	000001011 000000110 000000101 000000100 000000011 100000000 011100000 110010001 101010010	{67}
2738	9	1/3	3	{6, 5, 0, 7}	(0.857072, -2.61799)	-1.44547	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2739	9	1/3	3	{0, 7, 6, 5}	(0.857072, 2.61799)	-1.44547	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2740	9	1/3	3	{7, 0, 5, 6}	(0.857072, 0.523599)	1.44547	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2741	9	1/3	3	{5, 6, 7, 0}	(0.857072, -0.523599)	1.44547	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2742	9	1/3	3	{5, 6, 0, 7}	(1.28976, 2.61799)	2.24593	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2743	9	1/3	3	{0, 7, 5, 6}	(1.28976, -2.61799)	2.24593	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2744	9	1/3	3	{7, 0, 6, 5}	(1.28976, -0.523599)	-2.24593	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2745	9	1/3	3	{6, 5, 7, 0}	(1.28976, 0.523599)	-2.24593	000011001 000001111 000000111 000000001 100000010 110000001 011000000 011010001 111101010	{1}
2746	9	1/3	3	{0, 2, 1, 0}	(1.5708, -1.5708)	3.14159	000001011 000000110 000000101 000000100 000000011 100000000 011100000 110010001 101010010	{67}
2747	9	1/3	28	{2, 3, 2, 3}	(0, 0)	0	000001011 000001001 000000111 000000111 000000010 110000011 001100000 101111000 111101000	{64}
2748	9	1/3	28	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000001001 000000111 000000111 000000010 110000011 001100000 101111000 111101000	{64}
2749	9	1/3	26.5	{2, 3, 2, 3}	(0, 0)	0	000010111 000001100 000001011 000001011 100000111 011100010 110010001 101111001 101110110	{4}
2750	9	1/3	26.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001100 000001011 000001011 100000111 011100010 110010001 101111001 101110110	{4}
2751	9	1/3	26	{5, 6, 5, 6}	(0, 0)	0	000011100 000010011 000001110 000001110 110000011 101100001 101100001 011110000 010011100	{16}
2752	9	1/3	26	{5, 6, 6, 5}	(1.5708, 0)	-3.14159	000011100 000010011 000001110 000001110 110000011 101100001 101100001 011110000 010011100	{16}
2753	9	1/3	25	{0, 4, 3, 1}	(0, 0)	0	000101111 000011111 000000011 100001111 010001111 110110000 110110000 111110000 111110000	{16}
2754	9	1/3	25	{0, 4, 1, 3}	(1.5708, 0)	-3.14159	000101111 000011111 000000011 100001111 010001111 110110000 110110000 111110000 111110000	{16}
2755	9	1/3	24	{7, 8, 7, 8}	(0, 0)	0	000001111 000000111 000000111 000000110 000000101 100000011 111110011 111101101 111011110	{100}
2756	9	1/3	24	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001111 000000111 000000111 000000110 000000101 100000011 111110011 111101101 111011110	{100}
2757	9	1/3	22.5	{1, 2, 1, 2}	(0, 0)	0	000101111 000010111 000010111 100001111 011001011 100110011 111100001 111111000 111111100	{4}
2758	9	1/3	22.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000101111 000010111 000010111 100001111 011001011 100110011 111100001 111111000 111111100	{4}
2759	9	1/3	20.5	{1, 2, 1, 2}	(0, 0)	0	000010011 000001101 000001101 000000110 100000011 011000000 011100011 100110101 111010110	{8}
2760	9	1/3	20.5	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001101 000001101 000000110 100000011 011000000 011100011 100110101 111010110	{8}
2761	9	1/3	20	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000101 100000001 011000000 011100000 111000000 111110000	{56}
2762	9	1/3	20	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000101 100000001 011000000 011100000 111000000 111110000	{56}
2763	9	1/3	2.875	{5, 8, 6, 7}	(0, 0)	0	000101111 000010011 000000101 100001100 010000011 100100001 101100011 110010100 111011100	{8}
2764	9	1/3	2.875	{4, 2, 3, 5}	(0.857072, -1.5708)	-1.44547	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2765	9	1/3	2.875	{2, 4, 5, 3}	(0.857072, -1.5708)	1.44547	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2766	9	1/3	2.875	{5, 3, 2, 4}	(0.857072, 1.5708)	1.44547	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2767	9	1/3	2.875	{3, 5, 0, 1}	(0.857072, -2.61799)	-1.44547	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2768	9	1/3	2.875	{0, 1, 3, 5}	(0.857072, 2.61799)	-1.44547	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2769	9	1/3	2.875	{1, 0, 5, 3}	(0.857072, 0.523599)	1.44547	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2770	9	1/3	2.875	{3, 5, 4, 2}	(0.857072, 1.5708)	-1.44547	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2771	9	1/3	2.875	{5, 3, 1, 0}	(0.857072, -0.523599)	1.44547	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2772	9	1/3	2.875	{5, 3, 0, 1}	(1.28976, 2.61799)	2.24593	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2773	9	1/3	2.875	{3, 5, 2, 4}	(1.0472, -2.61799)	3.14159	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2774	9	1/3	2.875	{0, 1, 5, 3}	(1.28976, -2.61799)	2.24593	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2775	9	1/3	2.875	{1, 0, 3, 5}	(1.28976, -0.523599)	-2.24593	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2776	9	1/3	2.875	{3, 5, 1, 0}	(1.28976, 0.523599)	-2.24593	000011100 000010111 000001011 000000011 110001111 101010101 110011011 011110100 011111100	{1}
2777	9	1/3	2.875	{5, 3, 4, 2}	(1.0472, -0.523599)	-3.14159	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2778	9	1/3	2.875	{2, 4, 3, 5}	(1.0472, 2.61799)	3.14159	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2779	9	1/3	2.875	{4, 2, 5, 3}	(1.0472, 0.523599)	-3.14159	000101011 000011111 000010100 100000111 011001011 110010100 011101000 110110001 110110010	{1}
2780	9	1/3	2.875	{5, 8, 7, 6}	(1.5708, 0)	-3.14159	000101111 000010011 000000101 100001100 010000011 100100001 101100011 110010100 111011100	{8}
2781	9	1/3	2.85714	{0, 1, 4, 3}	(1.5708, 3.14159)	-1.42745	000100101 000010101 000001100 100000011 010000011 001000010 111000011 000111101 110110110	{4}
2782	9	1/3	2.85714	{0, 1, 3, 4}	(1.5708, -3.14159)	1.71414	000100101 000010101 000001100 100000011 010000011 001000010 111000011 000111101 110110110	{4}
2783	9	1/3	2.8	{2, 6, 7, 3}	(0, 0)	0	000010111 000001110 000000101 000000011 100000111 010000001 111010000 110110000 101111000	{8}
2784	9	1/3	2.8	{2, 6, 3, 7}	(1.5708, 0)	3.14159	000010111 000001110 000000101 000000011 100000111 010000001 111010000 110110000 101111000	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2785	9	1/3	2.75	{1, 8, 5, 4}	(0, 0)	0	000001100 000001000 000000110 000000011 000000001 110000011 101000001 001101000 000111100	{16}
2786	9	1/3	2.75	{1, 2, 5, 4}	(0.588003, -3.14159)	2.24593	000010111 000001011 000000111 000000001 100000001 010000001 101000001 111000000 111111100	{6}
2787	9	1/3	2.75	{8, 0, 7, 1}	(0.857072, -1.5708)	-1.44547	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{4}
2788	9	1/3	2.75	{1, 3, 5, 6}	(1.5708, 3.14159)	1.0472	000001010 000001001 000000110 000000101 000000011 110000101 001101001 101010001 010111110	{24}
2789	9	1/3	2.75	{4, 5, 0, 6}	(0.588003, -2.0944)	2.24593	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2790	9	1/3	2.75	{0, 3, 4, 7}	(0.523599, -1.5708)	-3.14159	000010111 000001011 000001010 000000101 100000001 011000101 100101000 111000000 110111000	{1}
2791	9	1/3	2.75	{5, 4, 6, 0}	(0.588003, -1.0472)	-2.24593	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2792	9	1/3	2.75	{2, 1, 4, 5}	(0.588003, 0)	-2.24593	000010111 000001011 000000111 000000001 100000001 010000001 101000001 111000000 111111100	{6}
2793	9	1/3	2.75	{0, 8, 1, 7}	(0.857072, -1.5708)	1.44547	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{4}
2794	9	1/3	2.75	{0, 6, 4, 5}	(0.588003, 2.0944)	2.24593	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2795	9	1/3	2.75	{1, 7, 0, 8}	(0.857072, 1.5708)	1.44547	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{4}
2796	9	1/3	2.75	{6, 0, 5, 3}	(0.588003, -3.14159)	-2.24593	000101110 000011010 000000011 100001101 010000101 110100011 100110011 111001101 001111110	{2}
2797	9	1/3	2.75	{4, 7, 0, 3}	(0.523599, 1.5708)	-3.14159	000010111 000001011 000001010 000000101 100000001 011000101 100101000 111000000 110111000	{1}
2798	9	1/3	2.75	{3, 4, 6, 7}	(1.5708, 0)	1.0472	000101100 000011010 000001111 100000111 010000111 111000001 101110000 011110000 001111000	{8}
2799	9	1/3	2.75	{7, 1, 8, 0}	(0.857072, 1.5708)	-1.44547	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{4}
2800	9	1/3	2.75	{6, 0, 5, 4}	(0.588003, 1.0472)	-2.24593	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2801	9	1/3	2.75	{0, 6, 3, 5}	(0.588003, 0)	2.24593	000101110 000011010 000000011 100001101 010000101 110100011 100110011 111001101 001111110	{2}
2802	9	1/3	2.75	{0, 6, 5, 3}	(1.5708, -1.0472)	-2.0944	000101110 000011010 000000011 100001101 010000101 110100011 100110011 111001101 001111110	{2}
2803	9	1/3	2.75	{7, 1, 0, 8}	(1.0472, -2.61799)	3.14159	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{4}
2804	9	1/3	2.75	{6, 0, 4, 5}	(1.10715, -2.0944)	-2.63623	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2805	9	1/3	2.75	{3, 4, 7, 6}	(1.5708, 0)	-2.0944	000101100 000011010 000001111 100000111 010000111 111000001 101110000 011110000 001111000	{8}
2806	9	1/3	2.75	{7, 4, 0, 3}	(1.0472, -1.5708)	3.14159	000010111 000001011 000001010 000000101 100000001 011000101 100101000 111000000 110111000	{1}
2807	9	1/3	2.75	{6, 0, 3, 5}	(1.5708, 1.0472)	-2.0944	000101110 000011010 000000011 100001101 010000101 110100011 100110011 111001101 001111110	{2}
2808	9	1/3	2.75	{0, 6, 5, 4}	(1.10715, -1.0472)	2.63623	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2809	9	1/3	2.75	{1, 7, 8, 0}	(1.0472, -0.523599)	-3.14159	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{2}
2810	9	1/3	2.75	{0, 8, 7, 1}	(1.0472, 2.61799)	3.14159	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{4}
2811	9	1/3	2.75	{2, 1, 5, 4}	(1.5708, -2.0944)	-2.0944	000010111 000001011 000000111 000000001 100000001 010000001 101000001 111000000 111111100	{6}
2812	9	1/3	2.75	{4, 5, 6, 0}	(1.10715, 2.0944)	-2.63623	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2813	9	1/3	2.75	{3, 0, 4, 7}	(1.0472, 1.5708)	-3.14159	000010111 000001011 000001010 000000101 100000001 011000101 100101000 111000000 110111000	{1}
2814	9	1/3	2.75	{5, 4, 0, 6}	(1.10715, 1.0472)	2.63623	000011100 000001101 000000111 000000011 100000011 110000000 111000000 001110000 011110000	{1}
2815	9	1/3	2.75	{1, 3, 6, 5}	(1.5708, 0)	2.0944	000001010 000001001 000000110 000000101 000000011 110000101 001101001 101010001 010111110	{24}
2816	9	1/3	2.75	{8, 0, 1, 7}	(1.0472, 0.523599)	-3.14159	000010111 000010100 000001111 000001010 110000001 001100001 111000000 101100000 101011000	{2}
2817	9	1/3	2.75	{1, 2, 4, 5}	(1.5708, 2.0944)	-2.0944	000010111 000001011 000000111 000000001 100000001 010000001 101000001 111000000 111111100	{6}
2818	9	1/3	2.75	{1, 8, 4, 5}	(1.5708, 0)	3.14159	000001100 000001000 000000110 000000011 000000001 110000011 101000001 001101000 000111100	{16}
2819	9	1/3	2.71429	{0, 1, 4, 5}	(1.5708, 3.14159)	-1.42745	000010101 000001101 000000111 000000011 100000010 010000010 111000001 001111000 111100100	{4}
2820	9	1/3	2.71429	{0, 1, 5, 4}	(1.5708, -3.14159)	1.71414	000010101 000001101 000000111 000000011 100000010 010000010 111000001 001111000 111100100	{4}
2821	9	1/3	2.66667	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001001 000000101 000000011 111000000 110100000 110010000 111110000	{28}
2822	9	1/3	2.66667	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001001 000000101 000000011 111000000 110100000 110010000 111100000	{28}
2823	9	1/3	2.625	{2, 3, 2, 3}	(0, 0)	0	000011001 000001111 000000111 000000111 100000001 110000000 011100010 011100100 111110000	{76}
2824	9	1/3	2.625	{1, 5, 2, 5}	(0, 0)	-3.14159	000101010 000010101 000000111 100001010 010000010 100100001 011000001 101110001 011001110	{8}
2825	9	1/3	2.625	{5, 4, 0, 6}	(0.857072, -2.61799)	-1.44547	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2826	9	1/3	2.625	{0, 6, 5, 4}	(0.857072, 2.61799)	-1.44547	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2827	9	1/3	2.625	{1, 5, 4, 2}	(1.5708, 0)	1.0472	000100111 000010110 000001101 100000111 010000010 001000001 111100000 110110001 101101010	{4}
2828	9	1/3	2.625	{6, 0, 4, 5}	(0.857072, 0.523599)	1.44547	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2829	9	1/3	2.625	{4, 5, 6, 0}	(0.857072, -0.523599)	1.44547	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2830	9	1/3	2.625	{4, 5, 0, 6}	(1.28976, 2.61799)	2.24593	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2831	9	1/3	2.625	{0, 6, 4, 5}	(1.28976, -2.61799)	2.24593	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2832	9	1/3	2.625	{1, 5, 2, 4}	(1.5708, 0)	-2.0944	000100111 000010110 000001101 100000111 010000010 001000001 111100000 110110001 101101010	{4}
2833	9	1/3	2.625	{6, 0, 5, 4}	(1.28976, -0.523599)	-2.24593	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2834	9	1/3	2.625	{5, 4, 6, 0}	(1.28976, 0.523599)	-2.24593	000110011 000011111 000001111 100000100 110000011 011000000 011100011 111010101 111010110	{1}
2835	9	1/3	2.625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011001 000001111 000000111 000000111 100000001 110000000 011100010 011100100 111110000	{76}
2836	9	1/3	2.625	{5, 1, 2, 5}	(1.5708, -1.5708)	3.14159	000101010 000010101 000000111 100001010 010000010 100100001 011000001 101110001 011001110	{8}
2837	9	1/3	2.6	{1, 5, 4, 2}	(0, 0)	0	000100101 000010111 000001111 100000010 010000000 001000000 111000001 011100000 111000100	{8}
2838	9	1/3	2.6	{1, 5, 2, 4}	(1.5708, 0)	3.14159	000100101 000010111 000001111 100000010 010000000 001000000 111000001 011100000 111000100	{8}
2839	9	1/3	2.57143	{2, 3, 6, 5}	(1.5708, 3.14159)	-1.42745	000010011 000001111 000001001 000000101 100000010 011000000 010100000 110010000 111100000	{4}
2840	9	1/3	2.57143	{2, 3, 5, 6}	(1.5708, -3.14159)	1.71414	000010011 000001111 000001001 000000101 100000010 011000000 010100000 110010000 111100000	{4}
2841	9	1/3	2.5	{0, 4, 5, 4}	(0, 0)	2.47465	000110111 000101111 000011011 110011111 101101111 011110101 110111011 111110100 111111100	{3}
2842	9	1/3	2.5	{7, 0, 7, 3}	(0, 0)	1.42745	000101111 000011101 000010011 100000001 011001111 110010111 110011011 101011101 111111110	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2843	9	1/3	2.5	{4, 0, 4, 5}	(0, 0)	-2.47465	000110111 000101111 000011011 110011111 101101111 011110101 110111011 111110100 111111100	{3}
2844	9	1/3	2.5	{0, 7, 3, 7}	(0, 0)	-1.42745	000101111 000011101 000010011 100000001 011001111 110010111 110011011 101011101 111111110	{4}
2845	9	1/3	2.5	{4, 0, 5, 4}	(1.5708, 1.23732)	-3.14159	000110111 000101111 000011011 110011111 101101111 011110101 110111011 111110100 111111100	{3}
2846	9	1/3	2.5	{0, 7, 7, 3}	(1.5708, 0.713724)	-3.14159	000101111 000011101 000010011 100000001 011001111 110010111 110011011 101011101 111111110	{4}
2847	9	1/3	2.5	{0, 4, 4, 5}	(1.5708, -1.23732)	3.14159	000110111 000101111 000011011 110011111 101101111 011110101 110111011 111110100 111111100	{3}
2848	9	1/3	2.5	{7, 0, 3, 7}	(1.5708, -0.713724)	3.14159	000101111 000011101 000010011 100000001 011001111 110010111 110011011 101011101 111111110	{4}
2849	9	1/3	2.46154	{0, 1, 2, 3}	(1.5708, 0)	-0.56207	000010110 000001110 000000101 000000011 100000101 010000011 111010001 110101001 001111110	{4}
2850	9	1/3	2.46154	{0, 1, 3, 2}	(1.5708, 0)	2.57952	000010110 000001110 000000101 000000011 100000101 010000011 111010001 110101001 001111110	{4}
2851	9	1/3	2.44444	{2, 3, 2, 3}	(0, 0)	0	000011111 000001101 000000011 000000011 100000011 110000110 110001001 101111000 111110100	{28}
2852	9	1/3	2.44444	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001101 000000011 000000011 100000011 110000110 110001001 101111000 111110100	{28}
2853	9	1/3	2.4	{7, 2, 7, 3}	(0, 0)	2.0944	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2854	9	1/3	2.4	{2, 8, 3, 7}	(0, 0)	1.0472	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2855	9	1/3	2.4	{2, 7, 3, 7}	(0, 0)	-2.0944	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2856	9	1/3	2.4	{8, 2, 7, 3}	(0, 0)	-1.0472	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2857	9	1/3	2.4	{2, 7, 7, 3}	(1.5708, 1.0472)	-3.14159	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2858	9	1/3	2.4	{8, 2, 3, 7}	(1.5708, 0.523599)	-3.14159	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2859	9	1/3	2.4	{7, 2, 3, 7}	(1.5708, -1.0472)	3.14159	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2860	9	1/3	2.4	{2, 8, 7, 3}	(1.5708, -0.523599)	3.14159	000010011 000001011 000000101 000000101 100000011 010000000 001100010 110010100 111110000	{4}
2861	9	1/3	2.375	{1, 2, 4, 3}	(0.588003, -3.14159)	2.24593	000101111 000010111 000001111 100000011 010000011 101000011 111000000 111111001 111111010	{2}
2862	9	1/3	2.375	{1, 2, 4, 5}	(1.5708, 3.14159)	1.0472	000100111 000010011 000001011 100000111 010000001 001000001 100100001 111100000 111111100	{8}
2863	9	1/3	2.375	{2, 1, 3, 4}	(0.588003, 0)	-2.24593	000101111 000010111 000001111 100000011 010000011 101000011 111000000 111111001 111111010	{2}
2864	9	1/3	2.375	{2, 1, 4, 3}	(1.5708, -2.0944)	-2.0944	000101111 000010111 000001111 100000011 010000011 101000011 111000000 111111001 111111010	{2}
2865	9	1/3	2.375	{1, 2, 5, 4}	(1.5708, 0)	2.0944	000100111 000010011 000001011 100000111 010000001 001000001 100100001 111100000 111111100	{8}
2866	9	1/3	2.375	{1, 2, 3, 4}	(1.5708, 2.0944)	-2.0944	000101111 000010111 000001111 100000011 010000011 101000011 111000000 111111001 111111010	{2}
2867	9	1/3	2.33333	{2, 7, 5, 6}	(0, 0)	2.0944	000100111 000010101 000001110 100000110 010000011 001000001 111100001 101110001 110011110	{16}
2868	9	1/3	2.33333	{7, 2, 6, 5}	(0, 0)	-2.0944	000100111 000010101 000001110 100000110 010000011 001000001 111100001 101110001 110011110	{16}
2869	9	1/3	2.33333	{7, 2, 5, 6}	(1.5708, 1.0472)	-3.14159	000100111 000010101 000001110 100000110 010000011 001000001 111100001 101110001 110011110	{16}
2870	9	1/3	2.33333	{2, 7, 6, 5}	(1.5708, -1.0472)	3.14159	000100111 000010101 000001110 100000110 010000011 001000001 111100001 101110001 110011110	{16}
2871	9	1/3	2.28571	{0, 1, 8, 6}	(1.5708, 3.14159)	-1.42745	000010110 000010011 000001110 000001011 110000101 001100000 101010001 111100000 010110100	{12}
2872	9	1/3	2.28571	{0, 1, 6, 8}	(1.5708, -3.14159)	1.71414	000010110 000010011 000001110 000001011 110000101 001100000 101010001 111100000 010110100	{12}
2873	9	1/3	2.25	{7, 2, 7, 3}	(0, 0)	2.0944	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2874	9	1/3	2.25	{2, 8, 3, 7}	(0, 0)	1.0472	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2875	9	1/3	2.25	{2, 7, 3, 7}	(0, 0)	-2.0944	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2876	9	1/3	2.25	{8, 2, 7, 3}	(0, 0)	-1.0472	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2877	9	1/3	2.25	{2, 3, 6, 5}	(0.588003, -3.14159)	2.24593	000001111 000001001 000000101 000000011 000000010 110000010 101000000 100111001 111100010	{6}
2878	9	1/3	2.25	{0, 1, 4, 5}	(1.5708, 3.14159)	1.0472	000010010 000001010 000000100 000000100 100000001 010000001 001100011 110000101 000011110	{16}
2879	9	1/3	2.25	{5, 6, 0, 7}	(0.588003, -2.0944)	2.24593	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2880	9	1/3	2.25	{6, 5, 7, 0}	(0.588003, -1.0472)	-2.24593	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2881	9	1/3	2.25	{3, 2, 5, 6}	(0.588003, 0)	-2.24593	000001111 000001001 000000101 000000011 000000010 110000010 101000000 100111001 111100010	{6}
2882	9	1/3	2.25	{0, 7, 5, 6}	(0.588003, 2.0944)	2.24593	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2883	9	1/3	2.25	{5, 6, 2, 8}	(0.857072, -2.61799)	-1.44547	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2884	9	1/3	2.25	{1, 0, 6, 5}	(1.5708, 1.0472)	1.0472	000011000 000001110 000000111 000000011 100000011 110000001 011000000 011110000 001111000	{2}
2885	9	1/3	2.25	{2, 8, 5, 6}	(0.857072, 2.61799)	-1.44547	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2886	9	1/3	2.25	{0, 2, 3, 1}	(1.5708, 3.14159)	-1.0472	000001010 000001001 000000110 000000101 000000011 110000011 001100011 101011101 010111110	{8}
2887	9	1/3	2.25	{8, 2, 6, 5}	(0.857072, 0.523599)	1.44547	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2888	9	1/3	2.25	{7, 0, 6, 5}	(0.588003, 1.0472)	-2.24593	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2889	9	1/3	2.25	{0, 1, 5, 6}	(1.5708, -1.0472)	1.0472	000011000 000001110 000000111 000000011 100000011 110000001 011000000 011110000 001111000	{2}
2890	9	1/3	2.25	{6, 5, 8, 2}	(0.857072, -0.523599)	1.44547	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2891	9	1/3	2.25	{0, 1, 6, 5}	(1.10715, 3.14159)	2.63623	000011000 000001110 000000111 000000011 100000011 110000001 011000000 011110000 001111000	{2}
2892	9	1/3	2.25	{6, 5, 2, 8}	(1.28976, 2.61799)	2.24593	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2893	9	1/3	2.25	{7, 0, 5, 6}	(1.10715, -2.0944)	-2.63623	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2894	9	1/3	2.25	{2, 8, 6, 5}	(1.28976, -2.61799)	2.24593	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2895	9	1/3	2.25	{0, 2, 1, 3}	(1.5708, 0)	-2.0944	000001010 000001001 000000110 000000101 000000011 110000011 001100011 101011101 010111110	{8}
2896	9	1/3	2.25	{8, 2, 5, 6}	(1.28976, -0.523599)	-2.24593	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2897	9	1/3	2.25	{5, 6, 8, 2}	(1.28976, 0.523599)	-2.24593	000001011 000001010 000000111 000000101 000000010 110000101 001101000 111010000 101101000	{1}
2898	9	1/3	2.25	{1, 0, 5, 6}	(1.10715, 0)	-2.63623	000011000 000001110 000000111 000000011 100000011 110000001 011000000 011110000 001111000	{2}
2899	9	1/3	2.25	{0, 7, 6, 5}	(1.10715, -1.0472)	2.63623	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2900	9	1/3	2.25	{3, 2, 6, 5}	(1.5708, -2.0944)	-2.0944	000001111 000001001 000000101 000000011 000000010 110000010 101000000 100111001 111100010	{6}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2901	9	1/3	2.25	{5, 6, 7, 0}	(1.10715, 2.0944)	-2.63623	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2902	9	1/3	2.25	{6, 5, 0, 7}	(1.10715, 1.0472)	2.63623	000001110 000000111 000000110 000000011 000000001 100000001 111000000 111100000 010111000	{1}
2903	9	1/3	2.25	{0, 1, 5, 4}	(1.5708, 0)	2.0944	000010010 000001010 000000100 000000100 100000001 010000001 001100011 110000101 000011110	{16}
2904	9	1/3	2.25	{2, 3, 5, 6}	(1.5708, 2.0944)	-2.0944	000001111 000001001 000000101 000000011 000000010 110000010 101000000 100111001 111100010	{6}
2905	9	1/3	2.25	{2, 7, 7, 3}	(1.5708, 1.0472)	-3.14159	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2906	9	1/3	2.25	{8, 2, 3, 7}	(1.5708, 0.523599)	-3.14159	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2907	9	1/3	2.25	{7, 2, 3, 7}	(1.5708, -1.0472)	3.14159	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2908	9	1/3	2.25	{2, 8, 7, 3}	(1.5708, -0.523599)	3.14159	000010011 000001011 000000101 000000101 100000011 010000011 001100010 110011100 111111000	{4}
2909	9	1/3	2.2	{0, 5, 4, 1}	(0, 0)	0	000010001 000001001 000000101 000000011 100000001 010000001 001000001 000100000 111111100	{24}
2910	9	1/3	2.2	{0, 5, 1, 4}	(1.5708, 0)	-3.14159	000010001 000001001 000000101 000000011 100000001 010000001 001000001 000100000 111111100	{24}
2911	9	1/3	2.14286	{0, 1, 4, 3}	(1.5708, 0)	-1.42745	000101010 000011010 000001111 100000101 010000101 111000011 001110011 111001100 001111100	{4}
2912	9	1/3	2.14286	{4, 0, 1, 3}	(0.785398, -3.14159)	-2.3664	000001110 000001010 000001001 000000111 000000101 111000001 100110000 110100000 001111000	{2}
2913	9	1/3	2.14286	{0, 4, 3, 1}	(0.785398, 0)	2.3664	000001110 000001010 000001001 000000111 000000101 111000001 100110000 110100000 001111000	{2}
2914	9	1/3	2.14286	{0, 4, 1, 3}	(1.5708, -1.0472)	-1.71414	000001110 000001010 000001001 000000111 000000101 111000001 100110000 110100000 001111000	{2}
2915	9	1/3	2.14286	{4, 0, 3, 1}	(1.5708, 1.0472)	-1.71414	000001110 000001010 000001001 000000111 000000101 111000001 100110000 110100000 001111000	{2}
2916	9	1/3	2.14286	{0, 1, 3, 4}	(1.5708, 0)	1.71414	000101010 000011010 000001111 100000101 010000101 111000011 001110011 111001100 001111100	{4}
2917	9	1/3	2.125	{2, 3, 2, 3}	(0, 0)	0	000001101 000000110 000000011 000000011 000000010 100000101 110001011 011110101 101101110	{64}
2918	9	1/3	2.125	{3, 2, 6, 1}	(0.588003, -2.0944)	2.24593	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2919	9	1/3	2.125	{2, 3, 1, 6}	(0.588003, -1.0472)	-2.24593	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2920	9	1/3	2.125	{6, 1, 3, 2}	(0.588003, 2.0944)	2.24593	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2921	9	1/3	2.125	{1, 5, 4, 2}	(1.5708, 0)	1.0472	000100111 000010110 000001100 100000111 010000011 001000001 111100001 110110000 100111100	{4}
2922	9	1/3	2.125	{1, 6, 2, 3}	(0.588003, 1.0472)	-2.24593	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2923	9	1/3	2.125	{1, 6, 3, 2}	(1.10715, -2.0944)	-2.63623	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2924	9	1/3	2.125	{1, 5, 2, 4}	(1.5708, 0)	-2.0944	000100111 000010110 000001100 100000111 010000011 001000001 111100001 110110000 100111100	{4}
2925	9	1/3	2.125	{6, 1, 2, 3}	(1.10715, -1.0472)	2.63623	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2926	9	1/3	2.125	{3, 2, 1, 6}	(1.10715, 2.0944)	-2.63623	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2927	9	1/3	2.125	{2, 3, 6, 1}	(1.10715, 1.0472)	2.63623	000111010 000011101 000001101 100000111 110000011 111000011 011100000 100111000 011111000	{1}
2928	9	1/3	2.125	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000000110 000000011 000000011 000000010 100000101 110001011 011110101 101101110	{64}
2929	9	1/3	2.1	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000111 000000011 110001011 110010011 111000000 111111001 111111010	{8}
2930	9	1/3	2.1	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000111 000000011 110001011 110010011 111000000 111111001 111111010	{8}
2931	9	1/3	2	{1, 8, 1, 4}	(0, 0)	1.0472	000001101 000001011 000000111 000000010 000000001 110000100 101001000 011100000 111010000	{90}
2932	9	1/3	2	{8, 1, 4, 1}	(0, 0)	-1.0472	000001101 000001011 000000111 000000010 000000001 110000100 101001000 011100000 111010000	{90}
2933	9	1/3	2	{8, 1, 1, 4}	(1.5708, 0.523599)	-3.14159	000001101 000001011 000000111 000000010 000000001 110000100 101001000 011100000 111010000	{90}
2934	9	1/3	2	{1, 8, 4, 1}	(1.5708, -0.523599)	3.14159	000001101 000001011 000000111 000000010 000000001 110000100 101001000 011100000 111010000	{90}
2935	9	1/3	18	{1, 2, 1, 2}	(0, 0)	0	000001011 000000100 000000100 000000010 000000010 100000001 011000001 100110001 100001110	{192}
2936	9	1/3	18	{0, 6, 2, 6}	(0, 0)	-3.14159	000010101 000001011 000000111 000000010 100000010 010000001 101000000 011110001 111001010	{6}
2937	9	1/3	18	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000100 000000100 000000010 000000010 100000001 011000001 100110001 100001110	{192}
2938	9	1/3	18	{6, 0, 2, 6}	(1.5708, -1.5708)	3.14159	000010101 000001011 000000111 000000010 100000010 010000001 101000000 011110001 111001010	{6}
2939	9	1/3	16.5	{2, 3, 2, 3}	(0, 0)	0	000011100 000001100 000000011 000000011 100000010 110000101 110001001 001110001 001101110	{24}
2940	9	1/3	16.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011100 000001100 000000011 000000011 100000010 110000101 110001001 001110001 001101110	{24}
2941	9	1/3	16	{6, 2, 6, 3}	(0, 0)	1.0472	000010101 000001110 000001011 000000001 100000010 011000111 110001001 011011000 101101100	{6}
2942	9	1/3	16	{2, 5, 3, 5}	(0, 0)	-3.14159	000011100 000001111 000000111 000000011 100000010 110000001 111000001 011110001 011101110	{6}
2943	9	1/3	16	{2, 6, 3, 6}	(0, 0)	-1.0472	000010101 000001110 000001011 000000001 100000010 011000111 110001001 011011000 101101100	{6}
2944	9	1/3	16	{2, 6, 6, 3}	(1.5708, 0.523599)	-3.14159	000010101 000001110 000001011 000000001 100000010 011000111 110001001 011011000 101101100	{6}
2945	9	1/3	16	{5, 2, 3, 5}	(1.5708, -1.5708)	3.14159	000011100 000001111 000000111 000000011 100000010 110000001 111000001 011110001 011101110	{6}
2946	9	1/3	16	{6, 2, 3, 6}	(1.5708, -0.523599)	3.14159	000010101 000001110 000001011 000000001 100000010 011000111 110001001 011011000 101101100	{6}
2947	9	1/3	14.5	{2, 3, 2, 3}	(0, 0)	0	000011101 000001110 000000011 000000011 100000000 110000111 110001011 011101101 101101110	{62}
2948	9	1/3	14.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000001110 000000011 000000011 100000000 110000111 110001011 011101101 101101110	{62}
2949	9	1/3	14	{1, 5, 2, 5}	(0, 0)	2.0944	000010111 000001100 000000011 000000001 100000110 010000011 110010001 101011000 101101100	{13}
2950	9	1/3	14	{7, 2, 7, 3}	(0, 0)	1.0472	000010100 000001110 000001001 000000011 100000011 011000101 110001001 010110000 001111100	{14}
2951	9	1/3	14	{5, 1, 5, 2}	(0, 0)	-2.0944	000010111 000001100 000000011 000000001 100000110 010000011 110010001 101011000 101101100	{13}
2952	9	1/3	14	{2, 7, 3, 7}	(0, 0)	-1.0472	000010100 000001110 000001001 000000011 100000011 011000101 110001001 010110000 001111100	{14}
2953	9	1/3	14	{5, 1, 2, 5}	(1.5708, 1.0472)	-3.14159	000010111 000001100 000000011 000000001 100000110 010000011 110010001 101011000 101101100	{13}
2954	9	1/3	14	{2, 7, 7, 3}	(1.5708, 0.523599)	-3.14159	000010100 000001110 000001001 000000011 100000011 011000101 110001001 010110000 001111100	{14}
2955	9	1/3	14	{1, 5, 5, 2}	(1.5708, -1.0472)	3.14159	000010111 000001100 000000011 000000001 100000110 010000011 110010001 101011000 101101100	{13}
2956	9	1/3	14	{7, 2, 3, 7}	(1.5708, -0.523599)	3.14159	000010100 000001110 000001001 000000011 100000011 011000101 110001001 010110000 001111100	{14}
2957	9	1/3	13	{1, 4, 5, 2}	(0, 0)	2.0944	000010101 000001001 000000111 000000011 100000010 010000001 101000001 001110000 111101100	{16}
2958	9	1/3	13	{7, 0, 1, 4}	(0, 0)	1.0472	000010011 000001111 000001001 000000101 100000011 011000011 010100000 110011000 111111000	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
2959	9	1/3	13	{4, 1, 2, 5}	(0, 0)	-2.0944	000010101 000001001 000000111 000000011 100000010 010000001 101000001 001110000 111101100	{16}
2960	9	1/3	13	{0, 7, 4, 1}	(0, 0)	-1.0472	000010011 000001111 000001001 000000101 100000011 011000011 010100000 110011000 111111000	{4}
2961	9	1/3	13	{4, 1, 5, 2}	(1.5708, 1.0472)	-3.14159	000010101 000001001 000000111 000000011 100000010 010000001 101000001 001110000 111101100	{16}
2962	9	1/3	13	{0, 7, 1, 4}	(1.5708, 0.523599)	-3.14159	000010011 000001111 000001001 000000101 100000011 011000011 010100000 110011000 111111000	{4}
2963	9	1/3	13	{1, 4, 2, 5}	(1.5708, -1.0472)	3.14159	000010101 000001001 000000111 000000011 100000010 010000001 101000001 001110000 111101100	{16}
2964	9	1/3	13	{7, 0, 4, 1}	(1.5708, -0.523599)	3.14159	000010011 000001111 000001001 000000101 100000011 011000011 010100000 110011000 111111000	{4}
2965	9	1/3	12	{1, 7, 8, 7}	(0, 0)	2.0944	000001111 000000111 000000011 000000011 000000011 100000001 110000010 111110101 111111010	{23}
2966	9	1/3	12	{3, 0, 3, 4}	(0, 0)	1.0472	000010101 000001111 000000110 000000011 100000011 010000001 111000000 011110001 110111010	{9}
2967	9	1/3	12	{3, 6, 5, 6}	(0, 0)	-3.14159	000010110 000001010 000000111 000000011 100000000 010000001 101000001 111100001 001101110	{14}
2968	9	1/3	12	{7, 1, 7, 8}	(0, 0)	-2.0944	000001111 000000111 000000011 000000011 000000011 100000001 110000010 111110101 111111010	{23}
2969	9	1/3	12	{0, 3, 4, 3}	(0, 0)	-1.0472	000010101 000001111 000000110 000000011 100000011 010000001 111000000 011110001 110111010	{9}
2970	9	1/3	12	{7, 1, 8, 7}	(1.5708, 1.0472)	-3.14159	000001111 000000111 000000011 000000011 000000011 100000001 110000010 111110101 111111010	{23}
2971	9	1/3	12	{0, 3, 3, 4}	(1.5708, 0.523599)	-3.14159	000010101 000001111 000000110 000000011 100000011 010000001 111000000 011110001 110111010	{9}
2972	9	1/3	12	{6, 3, 5, 6}	(1.5708, -1.5708)	3.14159	000010110 000001010 000000111 000000011 100000000 010000001 101000001 111100001 001101110	{14}
2973	9	1/3	12	{1, 7, 7, 8}	(1.5708, -1.0472)	3.14159	000001111 000000111 000000011 000000011 000000011 100000001 110000010 111110101 111111010	{23}
2974	9	1/3	12	{3, 0, 4, 3}	(1.5708, -0.523599)	3.14159	000010101 000001111 000000110 000000011 100000011 010000001 111000000 011110001 110111010	{9}
2975	9	1/3	11	{7, 2, 3, 4}	(0, 0)	2.0944	000010101 000001001 000000111 000000010 100000010 010000000 101000001 001110000 111000100	{8}
2976	9	1/3	11	{2, 7, 4, 3}	(0, 0)	-2.0944	000010101 000001001 000000111 000000010 100000010 010000000 101000001 001110000 111000100	{8}
2977	9	1/3	11	{2, 7, 3, 4}	(1.5708, 1.0472)	-3.14159	000010101 000001001 000000111 000000010 100000010 010000000 101000001 001110000 111000100	{8}
2978	9	1/3	11	{7, 2, 4, 3}	(1.5708, -1.0472)	3.14159	000010101 000001001 000000111 000000010 100000010 010000000 101000001 001110000 111000100	{8}
2979	9	1/3	10.5	{2, 3, 2, 3}	(0, 0)	0	000011011 000001000 000000111 000000111 100000011 110000000 001100000 101110001 101110010	{102}
2980	9	1/3	10.5	{1, 5, 2, 5}	(0, 0)	-3.14159	000010111 000001010 000001001 000000011 100000111 011000111 100011000 110111001 101111010	{10}
2981	9	1/3	10.5	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000001000 000000111 000000111 100000011 110000000 001100000 101110001 101110010	{102}
2982	9	1/3	10.5	{5, 1, 2, 5}	(1.5708, -1.5708)	3.14159	000010111 000001010 000001001 000000011 100000111 011000111 100011000 110111001 101111010	{10}
2983	9	1/3	10	{6, 2, 3, 2}	(0, 0)	2.0944	000010101 000001001 000000110 000000011 100000011 010000000 101000000 001110001 110110010	{43}
2984	9	1/3	10	{4, 0, 4, 7}	(0, 0)	1.0472	000101011 000010101 000001111 100000110 010000010 101000001 011100000 101110001 111001010	{9}
2985	9	1/3	10	{2, 6, 2, 3}	(0, 0)	-2.0944	000010101 000001001 000000110 000000011 100000011 010000000 101000000 001110001 110110010	{43}
2986	9	1/3	10	{0, 4, 7, 4}	(0, 0)	-1.0472	000101011 000010101 000001111 100000110 010000010 101000001 011100000 101110001 111001010	{9}
2987	9	1/3	10	{2, 6, 3, 2}	(1.5708, 1.0472)	-3.14159	000010101 000001001 000000110 000000011 100000011 010000000 101000000 001110001 110110010	{43}
2988	9	1/3	10	{0, 4, 4, 7}	(1.5708, 0.523599)	-3.14159	000101011 000010101 000001111 100000110 010000010 101000001 011100000 101110001 111001010	{9}
2989	9	1/3	10	{6, 2, 2, 3}	(1.5708, -1.0472)	3.14159	000010101 000001001 000000110 000000011 100000011 010000000 101000000 001110001 110110010	{43}
2990	9	1/3	10	{4, 0, 7, 4}	(1.5708, -0.523599)	3.14159	000101011 000010101 000001111 100000110 010000010 101000001 011100000 101110001 111001010	{9}
2991	9	1/3	1.875	{1, 8, 4, 7}	(0, 0)	2.0944	000100110 000010100 000001011 100000101 010000011 001000011 110100000 101011000 001111000	{8}
2992	9	1/3	1.875	{8, 1, 7, 4}	(0, 0)	-2.0944	000100110 000010100 000001011 100000101 010000011 001000011 110100000 101011000 001111000	{8}
2993	9	1/3	1.875	{1, 6, 4, 5}	(0.588003, -3.14159)	2.24593	000101111 000010010 000000101 100001111 010000001 100100001 101100010 110100101 101111010	{2}
2994	9	1/3	1.875	{1, 2, 4, 5}	(1.5708, 3.14159)	1.0472	000100111 000010010 000001010 100000111 010000001 001000001 100100000 111100001 100111010	{12}
2995	9	1/3	1.875	{4, 2, 7, 1}	(0.588003, -2.0944)	2.24593	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
2996	9	1/3	1.875	{2, 4, 1, 7}	(0.588003, -1.0472)	-2.24593	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
2997	9	1/3	1.875	{6, 1, 5, 4}	(0.588003, 0)	-2.24593	000101111 000010010 000000101 100001111 010000001 100100001 101100010 110100101 101111010	{2}
2998	9	1/3	1.875	{7, 1, 4, 2}	(0.588003, 2.0944)	2.24593	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
2999	9	1/3	1.875	{1, 5, 7, 4}	(0.857072, -2.61799)	-1.44547	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3000	9	1/3	1.875	{7, 4, 1, 5}	(0.857072, 2.61799)	-1.44547	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3001	9	1/3	1.875	{4, 7, 5, 1}	(0.857072, 0.523599)	1.44547	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3002	9	1/3	1.875	{1, 7, 2, 4}	(0.588003, 1.0472)	-2.24593	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
3003	9	1/3	1.875	{5, 1, 4, 7}	(0.857072, -0.523599)	1.44547	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3004	9	1/3	1.875	{5, 1, 7, 4}	(1.28976, 2.61799)	2.24593	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3005	9	1/3	1.875	{1, 7, 4, 2}	(1.10715, -2.0944)	-2.63623	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
3006	9	1/3	1.875	{7, 4, 5, 1}	(1.28976, -2.61799)	2.24593	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3007	9	1/3	1.875	{4, 7, 1, 5}	(1.28976, -0.523599)	-2.24593	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3008	9	1/3	1.875	{1, 5, 4, 7}	(1.28976, 0.523599)	-2.24593	000100111 000011101 000011000 100000111 011000011 011000010 110100000 100111000 110110000	{1}
3009	9	1/3	1.875	{7, 1, 2, 4}	(1.10715, -1.0472)	2.63623	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
3010	9	1/3	1.875	{6, 1, 4, 5}	(1.5708, -2.0944)	-2.0944	000101111 000010010 000000101 100001111 010000001 100100001 101100010 110100101 101111010	{2}
3011	9	1/3	1.875	{4, 2, 1, 7}	(1.10715, 2.0944)	-2.63623	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
3012	9	1/3	1.875	{2, 4, 7, 1}	(1.10715, 1.0472)	2.63623	000110101 000001111 000001011 100010101 100100010 011000000 110100000 011010000 111100000	{1}
3013	9	1/3	1.875	{1, 2, 5, 4}	(1.5708, 0)	2.0944	000100111 000010010 000001010 100000111 010000001 001000001 100100000 111100001 100111010	{12}
3014	9	1/3	1.875	{1, 6, 5, 4}	(1.5708, 2.0944)	-2.0944	000101111 000010010 000000101 100001111 010000001 100100001 101100010 110100101 101111010	{2}
3015	9	1/3	1.875	{8, 1, 4, 7}	(1.5708, 1.0472)	-3.14159	000100110 000010100 000001011 100000101 010000011 001000011 110100000 101011000 001111000	{8}
3016	9	1/3	1.875	{1, 8, 7, 4}	(1.5708, -1.0472)	3.14159	000100110 000010100 000001011 100000101 010000011 001000011 110100000 101011000 001111000	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3017	9	1/3	1.83333	{1, 2, 1, 2}	(0, 0)	0	000000110 000000011 000000011 000000010 000000001 000000001 100000001 111100001 011011110	{48}
3018	9	1/3	1.83333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000110 000000011 000000011 000000010 000000001 000000001 100000001 111100001 011011110	{48}
3019	9	1/3	1.68	{1, 2, 1, 2}	(0, 0)	0	000011111 000000111 000000111 000000101 100001011 100010000 111100001 111010000 111110100	{20}
3020	9	1/3	1.68	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000000111 000000111 000000101 100001011 100010000 111100001 111010000 111110100	{20}
3021	9	1/3	1.66667	{1, 2, 1, 2}	(0, 0)	0	000001100 000000111 000000111 000000010 000000001 100000011 111000000 011101000 011011000	{28}
3022	9	1/3	1.66667	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001100 000000111 000000111 000000010 000000001 100000011 111000000 011101000 011011000	{28}
3023	9	1/3	1.65625	{2, 3, 2, 3}	(0, 0)	0	000010101 000001110 000001011 000001011 100000101 011100010 110010011 011101101 101110110	{4}
3024	9	1/3	1.65625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001110 000001011 000001011 100000101 011100010 110010011 011101101 101110110	{4}
3025	9	1/3	1.625	{6, 2, 6, 5}	(0, 0)	1.0472	000100110 000010101 000001001 100000110 010000011 001000000 110100001 100110001 011010110	{30}
3026	9	1/3	1.625	{2, 6, 5, 6}	(0, 0)	-1.0472	000100110 000010101 000001001 100000110 010000011 001000000 110100001 100110001 011010110	{30}
3027	9	1/3	1.625	{2, 6, 6, 5}	(1.5708, 0.523599)	-3.14159	000100110 000010101 000001001 100000110 010000011 001000000 110100001 100110001 011010110	{30}
3028	9	1/3	1.625	{6, 2, 5, 6}	(1.5708, -0.523599)	3.14159	000100110 000010101 000001001 100000110 010000011 001000000 110100001 100110001 011010110	{30}
3029	9	1/3	1.61111	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000001001 100000111 011100011 111010010 111011100 111111000	{8}
3030	9	1/3	1.61111	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000001001 100000111 011100011 111010010 111011100 111111000	{8}
3031	9	1/3	1.58333	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000010 100000100 111000001 011010000 111100000 111001000	{8}
3032	9	1/3	1.58333	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000010 100000100 111000001 011010000 111100000 111001000	{8}
3033	9	1/3	1.55556	{6, 7, 6, 7}	(0, 0)	0	000100111 000010111 000001110 100000111 010000111 001000001 111110000 111110000 110111000	{4}
3034	9	1/3	1.55556	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000100111 000010111 000001110 100000111 010000111 001000001 111110000 111110000 110111000	{4}
3035	9	1/3	1.5	{8, 1, 8, 5}	(0, 0)	1.0472	000010111 000001111 000000101 000000011 100000111 010000000 111010000 110110000 111110000	{4}
3036	9	1/3	1.5	{1, 8, 5, 8}	(0, 0)	-1.0472	000010111 000001111 000000101 000000011 100000111 010000000 111010000 110110000 111110000	{4}
3037	9	1/3	1.5	{1, 8, 8, 5}	(1.5708, 0.523599)	-3.14159	000010111 000001111 000000101 000000011 100000111 010000000 111010000 110110000 111110000	{4}
3038	9	1/3	1.5	{8, 1, 5, 8}	(1.5708, -0.523599)	3.14159	000010111 000001111 000000101 000000011 100000111 010000000 111010000 110110000 111110000	{4}
3039	9	1/3	1.4	{1, 6, 5, 2}	(0, 0)	0	000010001 000001001 000000101 000000011 100000001 010000000 001000000 000100000 111110000	{24}
3040	9	1/3	1.4	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010001 000001001 000000101 000000011 100000001 010000000 001000000 000100000 111110000	{24}
3041	9	1/3	1.25	{3, 4, 3, 4}	(0, 0)	0	000001010 000000110 000000101 000000011 000000011 100000001 011000001 110110000 001111100	{40}
3042	9	1/3	1.25	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001010 000000110 000000101 000000011 000000011 100000001 011000001 110110000 001111100	{40}
3043	9	1/3	1.16667	{2, 3, 2, 3}	(0, 0)	0	000011111 000001011 000000111 000000111 100000111 110000011 101110000 111111001 111111010	{32}
3044	9	1/3	1.16667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001011 000000111 000000111 100000111 110000011 101110000 111111001 111111010	{32}
3045	9	1/3	1.12	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000110 100000001 111000001 111100011 111100100 111011100	{8}
3046	9	1/3	1.12	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000110 100000001 111000001 111100011 111100100 111011100	{8}
3047	9	1/3	1.1	{1, 2, 1, 2}	(0, 0)	0	000000110 000000011 000000011 000000001 000000001 000000001 100000001 111000001 011111110	{8}
3048	9	1/3	1.1	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000000110 000000011 000000011 000000001 000000001 000000001 100000001 111000001 011111110	{8}
3049	9	1/3	1.05882	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000111 000000111 000000011 110000011 111100000 111111001 111111010	{8}
3050	9	1/3	1.05882	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000111 000000111 000000011 110000011 111100000 111111001 111111010	{8}
3051	9	1/3	1.05556	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000000111 100000010 111000000 111100001 011110000 111100100	{4}
3052	9	1/3	1.05556	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000000111 100000010 111000000 111100001 011110000 111100100	{4}
3053	9	1/3	0.941176	{1, 2, 1, 2}	(0, 0)	0	000001100 000000111 000000111 000000011 000000011 100000011 111000011 011111101 011111110	{8}
3054	9	1/3	0.941176	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001100 000000111 000000111 000000011 000000011 100000011 111000011 011111101 011111110	{8}
3055	9	1/3	0.933333	{2, 3, 2, 3}	(0, 0)	0	000001010 000000111 000000011 000000011 000000001 100000001 010000000 111100000 011111000	{8}
3056	9	1/3	0.933333	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001010 000000111 000000011 000000011 000000001 100000001 010000000 111100000 011111000	{8}
3057	9	1/3	0.888889	{2, 3, 2, 3}	(0, 0)	0	000010101 000001111 000000111 000000111 100000010 010000000 111100001 011110000 111100100	{4}
3058	9	1/3	0.888889	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001111 000000111 000000111 100000010 010000000 111100001 011110000 111100100	{4}
3059	9	1/3	0.88	{3, 4, 3, 4}	(0, 0)	0	000001111 000000101 000000101 000000011 000000011 100000011 111000001 100111000 111111100	{4}
3060	9	1/3	0.88	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000101 000000101 000000011 000000011 100000011 111000001 100111000 111111100	{4}
3061	9	1/3	0.777778	{2, 3, 2, 3}	(0, 0)	0	000011111 000001010 000000111 000000111 100000111 110000011 101110001 111111000 101111100	{12}
3062	9	1/3	0.777778	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001010 000000111 000000111 100000111 110000011 101110001 111111000 101111100	{12}
3063	9	1/3	0.75	{2, 3, 2, 3}	(0, 0)	0	000001111 000000101 000000011 000000011 000000010 100000011 110000001 101111000 111101100	{64}
3064	9	1/3	0.75	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001111 000000101 000000011 000000011 000000010 100000011 110000001 101111000 111101100	{64}
3065	9	1/3	0.72	{3, 4, 3, 4}	(0, 0)	0	000001111 000000011 000000011 000000001 000000001 100000110 100001000 111001001 111110010	{4}
3066	9	1/3	0.72	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000000011 000000011 000000001 000000001 100000110 100001000 111001001 111110010	{4}
3067	9	1/3	0.666667	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000010 000000010 000000001 000000001 110000011 001100100 110011100	{32}
3068	9	1/3	0.666667	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000010 000000010 000000001 000000001 110000011 001100100 110011100	{32}
3069	9	1/3	0.65625	{2, 3, 2, 3}	(0, 0)	0	000011111 000001001 000000111 000000111 100000111 110000001 101110010 101110100 111111000	{12}
3070	9	1/3	0.65625	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000001001 000000111 000000111 100000111 110000001 101110010 101110100 111111000	{12}
3071	9	1/3	0.48	{1, 2, 1, 2}	(0, 0)	0	000001111 000000111 000000111 000000011 000000011 100000101 111001000 111110001 111111010	{4}
3072	9	1/3	0.48	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001111 000000111 000000111 000000011 000000011 100000101 111001000 111110001 111111010	{4}
3073	9	1/3	0.470588	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000011 100001111 111010110 111011001 111111001 111110110	{8}
3074	9	1/3	0.470588	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000011 100001111 111010110 111011001 111111001 111110110	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3075	9	1/3	0.388889	{2, 3, 2, 3}	(0, 0)	0	000001011 000000110 000000011 000000011 000000001 100000011 010000001 111101000 101111100	{4}
3076	9	1/3	0.388889	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001011 000000110 000000011 000000011 000000001 100000011 010000001 111101000 101111100	{4}
3077	9	1/3	0.380952	{2, 3, 2, 3}	(0, 0)	0	000010111 000001101 000000111 000000111 100000111 010000010 111110001 101111000 111110100	{8}
3078	9	1/3	0.380952	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001101 000000111 000000111 100000111 010000010 111110001 101111000 111110100	{8}
3079	9	1/3	0.333333	{3, 4, 3, 4}	(0, 0)	0	000001001 000000101 000000011 000000001 000000001 100000001 010000000 001000000 111111000	{4}
3080	9	1/3	0.333333	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000101 000000011 000000001 000000001 100000001 010000000 001000000 111111000	{4}
3081	9	1/3	0.222222	{3, 4, 3, 4}	(0, 0)	0	000001001 000000101 000000011 000000001 000000001 100000001 010000001 001000000 111111100	{4}
3082	9	1/3	0.222222	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000101 000000011 000000001 000000001 100000001 010000001 001000000 111111100	{4}
3083	9	1/3	0.166667	{3, 4, 3, 4}	(0, 0)	0	000001001 000000101 000000011 000000001 000000001 100000001 010000001 001000001 111111110	{4}
3084	9	1/3	0.166667	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000101 000000011 000000001 000000001 100000001 010000001 001000001 111111110	{4}
3085	9	1/4	8.82843	{2, 3, 2, 3}	(0, 0)	0	000010101 000001001 000000011 000000011 100000101 010000001 100010000 001100000 111111000	{20}
3086	9	1/4	8.82843	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001001 000000011 000000011 100000101 010000001 100010000 001100000 111111000	{20}
3087	9	1/4	8.68629	{2, 3, 2, 3}	(0, 0)	0	000001101 000001011 000000110 000000110 000000011 110000101 101101001 011110001 110011110	{12}
3088	9	1/4	8.68629	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001011 000000110 000000110 000000011 110000101 101101001 011110001 110011110	{12}
3089	9	1/4	60.2843	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000111 000000100 110001011 110010011 111100001 111011001 111011110	{8}
3090	9	1/4	60.2843	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000111 000000100 110001011 110010011 111100001 111011001 111011110	{8}
3091	9	1/4	6.68629	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000111 000000101 110000011 110000011 111100001 111011001 111111110	{4}
3092	9	1/4	6.68629	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000111 000000101 110000011 110000011 111100001 111011001 111111110	{4}
3093	9	1/4	6.34315	{3, 4, 3, 4}	(0, 0)	0	000001010 000001001 000000110 000000101 000000101 110000011 001110010 101001101 010111010	{24}
3094	9	1/4	6.34315	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001010 000001001 000000110 000000101 000000101 110000011 001110010 101001101 010111010	{24}
3095	9	1/4	6	{3, 8, 7, 8}	(0, 0)	2.35619	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
3096	9	1/4	6	{8, 3, 8, 7}	(0, 0)	-2.35619	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
3097	9	1/4	6	{8, 3, 7, 8}	(1.5708, 1.1781)	-3.14159	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
3098	9	1/4	6	{3, 8, 8, 7}	(1.5708, -1.1781)	3.14159	000010011 000001111 000001111 000000101 100000000 011000000 011100000 111000000 111100000	{4}
3099	9	1/4	50.6274	{3, 4, 3, 4}	(0, 0)	0	000001101 000001011 000001010 000000110 000000110 111000001 100110001 011110000 110001100	{12}
3100	9	1/4	50.6274	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001101 000001011 000001010 000000110 000000110 111000001 100110001 011110000 110001100	{12}
3101	9	1/4	46.6274	{2, 3, 2, 3}	(0, 0)	0	000010110 000001111 000000001 000000001 100000101 010000010 110010000 110001000 011110000	{8}
3102	9	1/4	46.6274	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010110 000001111 000000001 000000001 100000101 010000010 110010000 110001000 011110000	{8}
3103	9	1/4	4.74755	{7, 0, 6, 1}	(1.5708, -1.5708)	-1.23096	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
3104	9	1/4	4.74755	{0, 7, 1, 6}	(1.5708, 1.5708)	-1.23096	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
3105	9	1/4	4.74755	{7, 0, 1, 6}	(0.955317, 0)	-3.14159	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
3106	9	1/4	4.74755	{0, 7, 6, 1}	(0.955317, -3.14159)	-3.14159	000110111 000101111 000011010 110000110 101000111 011000001 110110001 111110000 110011100	{2}
3107	9	1/4	4	{2, 8, 7, 8}	(0, 0)	2.35619	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
3108	9	1/4	4	{7, 1, 7, 6}	(0, 0)	0.785398	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
3109	9	1/4	4	{8, 2, 8, 7}	(0, 0)	-2.35619	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
3110	9	1/4	4	{1, 7, 6, 7}	(0, 0)	-0.785398	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
3111	9	1/4	4	{8, 2, 7, 8}	(1.5708, 1.1781)	-3.14159	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
3112	9	1/4	4	{1, 7, 7, 6}	(1.5708, 0.392699)	-3.14159	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
3113	9	1/4	4	{2, 8, 8, 7}	(1.5708, -1.1781)	3.14159	000001111 000001111 000000101 000000011 000000011 110000000 111000000 110110000 111110000	{8}
3114	9	1/4	4	{7, 1, 6, 7}	(1.5708, -0.392699)	3.14159	000001001 000000100 000000100 000000011 000000011 100000000 011000011 000110100 100110100	{8}
3115	9	1/4	38.9706	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000011010 000000111 111000011 111000011 110100001 111111001 110111110	{4}
3116	9	1/4	38.9706	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000011010 000000111 111000011 111000011 110100001 111111001 110111110	{4}
3117	9	1/4	36.9706	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000100 101001001 110110000 101110100	{16}
3118	9	1/4	36.9706	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000100 101001001 110110000 101110100	{16}
3119	9	1/4	3.54692	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000011 110000000 101100001 011110000 101110100	{4}
3120	9	1/4	3.54692	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000011 110000000 101100001 011110000 101110100	{4}
3121	9	1/4	3.43146	{2, 3, 2, 3}	(0, 0)	0	000011110 000010001 000001111 000001111 110000011 101100010 101100001 101111000 011110100	{8}
3122	9	1/4	3.43146	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000010001 000001111 000001111 110000011 101100010 101100001 101111000 011110100	{8}
3123	9	1/4	3.17157	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000111 101001001 110111000 101111100	{44}
3124	9	1/4	3.17157	{0, 1, 3, 4}	(1.5708, 3.14159)	1.5708	000100101 000010101 000001110 100000011 010000011 001000000 111000000 001110000 110110000	{4}
3125	9	1/4	3.17157	{0, 1, 4, 3}	(1.5708, 0)	1.5708	000100101 000010101 000001110 100000011 010000011 001000000 111000000 001110000 110110000	{4}
3126	9	1/4	3.17157	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000111 101001001 110111000 101111100	{44}
3127	9	1/4	3.02944	{3, 4, 3, 4}	(0, 0)	0	000001100 000001010 000000101 000000011 000000011 110000011 101000001 010111001 001111110	{24}
3128	9	1/4	3.02944	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001100 000001010 000000101 000000011 000000011 110000011 101000001 010111001 001111110	{24}
3129	9	1/4	3	{7, 0, 2, 1}	(0, 0)	2.35619	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3130	9	1/4	3	{0, 7, 0, 2}	(0, 0)	0.785398	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3131	9	1/4	3	{0, 7, 1, 2}	(0, 0)	-2.35619	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3132	9	1/4	3	{7, 0, 2, 0}	(0, 0)	-0.785398	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3133	9	1/4	3	{0, 7, 2, 1}	(1.5708, 1.1781)	-3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3134	9	1/4	3	{7, 0, 0, 2}	(1.5708, 0.392699)	-3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3135	9	1/4	3	{7, 0, 1, 2}	(1.5708, -1.1781)	3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3136	9	1/4	3	{0, 7, 2, 0}	(1.5708, -0.392699)	3.14159	000001011 000000111 000000010 000000001 000000001 100000000 010000000 111000000 110110000	{4}
3137	9	1/4	29.6569	{6, 7, 6, 7}	(0, 0)	0	000101111 000010111 000010110 100001111 011001111 100110111 111111001 111111001 110111110	{4}
3138	9	1/4	29.6569	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000101111 000010111 000010110 100001111 011001111 100110111 111111001 111111001 110111110	{4}
3139	9	1/4	25.3137	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001001 000000111 000000111 111000011 110110010 110111101 111111010	{8}
3140	9	1/4	25.3137	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001001 000000111 000000111 111000011 110110010 110111101 111111010	{8}
3141	9	1/4	23.3137	{2, 3, 2, 3}	(0, 0)	0	000010111 000001110 000000001 000000001 100000100 010000011 110010001 110001000 101101100	{4}
3142	9	1/4	23.3137	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001110 000000001 000000001 100000100 010000011 110010001 110001000 101101100	{4}
3143	9	1/4	21.6569	{2, 3, 2, 3}	(0, 0)	0	000010111 000001101 000001011 000001011 100000110 011100111 110011001 101111000 111101100	{36}
3144	9	1/4	21.6569	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001101 000001011 000001011 100000110 011100111 110011001 101111000 111101100	{36}
3145	9	1/4	20	{2, 3, 2, 3}	(0, 0)	0	000011000 000010111 000001111 000001111 110000011 101100010 011100001 011111000 011110100	{8}
3146	9	1/4	20	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011000 000010111 000001111 000001111 110000011 101100010 011100001 011111000 011110100	{8}
3147	9	1/4	2.86193	{0, 3, 1, 5}	(1.5708, -1.5708)	-1.23096	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
3148	9	1/4	2.86193	{3, 0, 5, 1}	(1.5708, 1.5708)	-1.23096	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
3149	9	1/4	2.86193	{0, 3, 5, 1}	(0.955317, 0)	-3.14159	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
3150	9	1/4	2.86193	{3, 0, 1, 5}	(0.955317, -3.14159)	-3.14159	000111101 000101111 000011011 110000111 101000101 111000011 110110011 011101101 111111110	{2}
3151	9	1/4	2.66667	{2, 3, 2, 3}	(0, 0)	0	000000110 000000101 000000011 000000011 000000001 000000001 110000000 101100000 011111000	{4}
3152	9	1/4	2.66667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000110 000000101 000000011 000000011 000000001 000000001 110000000 101100000 011111000	{4}
3153	9	1/4	2.58579	{0, 1, 4, 5}	(1.5708, 3.14159)	1.5708	000010101 000001101 000000110 000000110 100000011 010000011 111100000 001111000 110011000	{4}
3154	9	1/4	2.58579	{0, 1, 5, 4}	(1.5708, 0)	1.5708	000010101 000001101 000000110 000000110 100000011 010000011 111100000 001111000 110011000	{4}
3155	9	1/4	2.4	{2, 3, 2, 3}	(0, 0)	0	000001100 000001011 000000111 000000111 000000011 110000000 101100000 011110000 011110000	{4}
3156	9	1/4	2.4	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001100 000001011 000000111 000000111 000000011 110000000 101100000 011110000 011110000	{4}
3157	9	1/4	2.29289	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000001 110000000 101100001 011100000 101110100	{8}
3158	9	1/4	2.29289	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000001 110000000 101100001 011100000 101110100	{8}
3159	9	1/4	2.25548	{1, 2, 1, 2}	(0, 0)	0	000001011 000000111 000000111 000000010 000000010 100000001 011000001 111110000 111001100	{4}
3160	9	1/4	2.25548	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000111 000000111 000000010 000000010 100000001 011000001 111110000 111001100	{4}
3161	9	1/4	2.20349	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000000110 100000001 011000001 111100000 111100000 111011000	{4}
3162	9	1/4	2.20349	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000000110 100000001 011000001 111100000 111100000 111011000	{4}
3163	9	1/4	2.17157	{0, 8, 5, 7}	(0, 0)	2.35619	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
3164	9	1/4	2.17157	{7, 0, 7, 5}	(0, 0)	0.785398	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
3165	9	1/4	2.17157	{8, 0, 7, 5}	(0, 0)	-2.35619	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
3166	9	1/4	2.17157	{0, 7, 5, 7}	(0, 0)	-0.785398	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
3167	9	1/4	2.17157	{8, 0, 5, 7}	(1.5708, 1.1781)	-3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
3168	9	1/4	2.17157	{0, 7, 7, 5}	(1.5708, 0.392699)	-3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
3169	9	1/4	2.17157	{0, 8, 7, 5}	(1.5708, -1.1781)	3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{4}
3170	9	1/4	2.17157	{7, 0, 5, 7}	(1.5708, -0.392699)	3.14159	000001011 000000111 000000111 000000010 000000001 100000011 011000011 111101100 111011100	{8}
3171	9	1/4	2.15802	{2, 3, 2, 3}	(0, 0)	0	000011101 000010010 000001111 000001111 110000000 101100001 101100001 011100000 101101100	{4}
3172	9	1/4	2.15802	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011101 000010010 000001111 000001111 110000000 101100001 101100001 011100000 101101100	{4}
3173	9	1/4	2.11438	{4, 5, 4, 5}	(0, 0)	0	000011101 000011101 000011010 000000110 111000011 111000011 110100011 001111100 110011100	{4}
3174	9	1/4	2.11438	{4, 5, 5, 4}	(1.5708, 0)	3.14159	000011101 000011101 000011010 000000110 111000011 111000011 110100011 001111100 110011100	{4}
3175	9	1/4	2	{0, 4, 5, 3}	(0, 0)	0.785398	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
3176	9	1/4	2	{4, 0, 3, 5}	(0, 0)	-0.785398	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
3177	9	1/4	2	{4, 0, 5, 3}	(1.5708, 0.392699)	-3.14159	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
3178	9	1/4	2	{0, 4, 3, 5}	(1.5708, -0.392699)	3.14159	000001001 000000101 000000101 000000010 000000010 100000000 011000000 000110000 111000000	{8}
3179	9	1/4	17.6569	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000000 101000001 110110001 101110110	{20}
3180	9	1/4	17.6569	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000000 101000001 110110001 101110110	{20}
3181	9	1/4	16	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001011 000000101 110000110 111000000 110110000 111010001 111100010	{16}
3182	9	1/4	16	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001011 000000101 110000110 111000000 110110000 111010001 111100010	{16}
3183	9	1/4	10.3431	{2, 3, 2, 3}	(0, 0)	0	000011011 000001001 000000111 000000111 100000010 110000001 001100011 101110100 111101100	{56}
3184	9	1/4	10.3431	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000001001 000000111 000000111 100000010 110000001 001100011 101110100 111101100	{56}
3185	9	1/4	1.91037	{1, 2, 1, 2}	(0, 0)	0	000001011 000000111 000000111 000000101 000000010 100000001 011100001 111010000 111101100	{4}
3186	9	1/4	1.91037	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001011 000000111 000000111 000000101 000000010 100000001 011100001 111010000 111101100	{4}
3187	9	1/4	1.7746	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000000110 110000001 111000111 111101011 111101101 111011110	{12}
3188	9	1/4	1.7746	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000000110 110000001 111000111 111101011 111101101 111011110	{12}
3189	9	1/4	1.71108	{1, 2, 1, 2}	(0, 0)	0	000011101 000001111 000001111 000000011 100000001 111000000 111000000 011100001 111110010	{4}
3190	9	1/4	1.71108	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011101 000001111 000001111 000000011 100000001 111000000 111000000 011100001 111110010	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3191	9	1/4	1.67157	{1, 2, 4, 5}	(1.5708, -3.14159)	0.785398	000100111 000010010 000001010 100000110 010000001 001000001 100100001 111100001 100011110	{4}
3192	9	1/4	1.67157	{1, 2, 5, 4}	(1.5708, 3.14159)	-2.35619	000100111 000010010 000001010 100000110 010000001 001000001 100100001 111100001 100011110	{4}
3193	9	1/4	1.58579	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000011 100000001 111000111 111001000 111101001 111111010	{8}
3194	9	1/4	1.58579	{1, 2, 2, 1}	(1.5708, 0)	3.14159	000011111 000001111 000001111 000000011 100000001 111000111 111001000 111101001 111111010	{8}
3195	9	1/4	1.51472	{3, 4, 3, 4}	(0, 0)	0	000001111 000001010 000000101 000000011 000000011 110000011 101000001 110111001 101111110	{24}
3196	9	1/4	1.51472	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001111 000001010 000000101 000000011 000000011 110000011 101000001 110111001 101111110	{24}
3197	9	1/4	1.47759	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000100 000000011 000000011 100000001 011000000 010110001 110111010	{4}
3198	9	1/4	1.47759	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000100 000000011 000000011 100000001 011000000 010110001 110111010	{4}
3199	9	1/4	1.35925	{3, 4, 3, 4}	(0, 0)	0	000001011 000000110 000000101 000000011 000000011 100000001 011000000 110110000 101111000	{8}
3200	9	1/4	1.35925	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001011 000000110 000000101 000000011 000000011 100000001 011000000 110110000 101111000	{8}
3201	9	1/4	1.33333	{4, 5, 4, 5}	(0, 0)	0	000000100 000000100 000000011 000000010 000000001 000000001 110000001 001100000 001011100	{4}
3202	9	1/4	1.33333	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000100 000000100 000000011 000000010 000000001 000000001 110000001 001100000 001011100	{4}
3203	9	1/4	1.20468	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000011 100000111 111000110 011011001 111111000 111110100	{4}
3204	9	1/4	1.20468	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000011 100000111 111000110 011011001 111111000 111110100	{4}
3205	9	1/4	1.08831	{1, 2, 1, 2}	(0, 0)	0	000011011 000001111 000001111 000000011 100000101 111000010 011010000 111101001 111110010	{8}
3206	9	1/4	1.08831	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011011 000001111 000001111 000000011 100000101 111000010 011010000 111101001 111110010	{8}
3207	9	1/4	1.07901	{1, 2, 1, 2}	(0, 0)	0	000001001 000000101 000000101 000000010 000000010 100000001 011000011 000110101 111001110	{4}
3208	9	1/4	1.07901	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000001001 000000101 000000101 000000010 000000010 100000001 011000011 000110101 111001110	{4}
3209	9	1/4	1.00981	{1, 2, 1, 2}	(0, 0)	0	000001101 000000111 000000111 000000011 000000011 100000001 111000000 011110001 111111010	{4}
3210	9	1/4	1.00981	{1, 2, 2, 1}	(1.5708, 0)	3.14159	000001101 000000111 000000111 000000011 000000011 100000001 111000000 011110001 111111010	{4}
3211	9	1/4	0.8	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000000 001100000 111011000	{4}
3212	9	1/4	0.8	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000000 001100000 111011000	{4}
3213	9	1/4	0.757359	{2, 3, 2, 3}	(0, 0)	0	000010111 000001001 000000011 000000011 100000110 010000001 100010010 101110100 111101000	{8}
3214	9	1/4	0.757359	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010111 000001001 000000011 000000011 100000110 010000001 100010010 101110100 111101000	{8}
3215	9	1/4	0.700881	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000111 000000011 000000011 100000001 011000000 011110001 111111010	{4}
3216	9	1/4	0.700881	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000111 000000011 000000011 100000001 011000000 011110001 111111010	{4}
3217	9	1/4	0.671573	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000011 110000111 101101001 011111000 101111100	{12}
3218	9	1/4	0.671573	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000011 110000111 101101001 011111000 101111100	{12}
3219	9	1/4	0.666667	{4, 5, 4, 5}	(0, 0)	0	000000100 000000100 000000011 000000011 000000001 000000001 110000001 001100000 001111100	{4}
3220	9	1/4	0.666667	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000100 000000100 000000011 000000011 000000001 000000001 110000001 001100000 001111100	{4}
3221	9	1/4	0.649165	{3, 4, 3, 4}	(0, 0)	0	000001001 000000110 000000101 000000011 000000011 100000000 011000011 010110100 101110100	{4}
3222	9	1/4	0.649165	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000110 000000101 000000011 000000011 100000000 011000011 010110100 101110100	{4}
3223	9	1/4	0.632069	{2, 3, 2, 3}	(0, 0)	0	000000110 000000101 000000011 000000011 000000001 000000001 110000011 101100100 011111100	{4}
3224	9	1/4	0.632069	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000000110 000000101 000000011 000000011 000000001 000000001 110000011 101100100 011111100	{4}
3225	9	1/4	0.615849	{0, 1, 0, 1}	(0, 0)	0	000001110 000001110 000001001 000000111 000000110 111000001 110110001 110110001 001101110	{4}
3226	9	1/4	0.615849	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001110 000001110 000001001 000000111 000000110 111000001 110110001 110110001 001101110	{4}
3227	9	1/4	0.608555	{2, 3, 2, 3}	(0, 0)	0	000001101 000001010 000000111 000000111 000000001 110000111 101101001 011101000 101111100	{4}
3228	9	1/4	0.608555	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001101 000001010 000000111 000000111 000000001 110000111 101101001 011101000 101111100	{4}
3229	9	1/4	0.598625	{0, 1, 0, 1}	(0, 0)	0	000011101 000011101 000011011 000000110 111000011 111000011 110100010 001111101 111011010	{4}
3230	9	1/4	0.598625	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011101 000011101 000011011 000000110 111000011 111000011 110100010 001111101 111011010	{4}
3231	9	1/4	0.585786	{4, 5, 4, 5}	(0, 0)	0	000000101 000000100 000000011 000000010 000000001 000000001 110000001 001100000 101011100	{4}
3232	9	1/4	0.585786	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000100 000000011 000000010 000000001 000000001 110000001 001100000 101011100	{4}
3233	9	1/4	0.530818	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000100 000000011 000000011 100000001 011000011 010110101 110111110	{4}
3234	9	1/4	0.530818	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000100 000000011 000000011 100000001 011000011 010110101 110111110	{4}
3235	9	1/4	0.514719	{3, 4, 3, 4}	(0, 0)	0	000001011 000000110 000000101 000000011 000000011 100000001 011000011 110110100 101111100	{4}
3236	9	1/4	0.514719	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001011 000000110 000000101 000000011 000000011 100000001 011000011 110110100 101111100	{4}
3237	9	1/4	0.510958	{4, 5, 4, 5}	(0, 0)	0	000000101 000000100 000000010 000000010 000000001 000000001 110000001 001100001 100011110	{4}
3238	9	1/4	0.510958	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000100 000000010 000000010 000000001 000000001 110000001 001100001 100011110	{4}
3239	9	1/4	0.5	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000000 001100000 111111000	{4}
3240	9	1/4	0.5	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000000 001100000 111111000	{4}
3241	9	1/4	0.406983	{4, 5, 4, 5}	(0, 0)	0	000000101 000000100 000000011 000000011 000000001 000000001 110000001 001100000 101111100	{4}
3242	9	1/4	0.406983	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000100 000000011 000000011 000000001 000000001 110000001 001100000 101111100	{4}
3243	9	1/4	0.316034	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100000 111011100	{4}
3244	9	1/4	0.316034	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100000 111011100	{4}
3245	9	1/4	0.299312	{3, 4, 3, 4}	(0, 0)	0	000001001 000000111 000000111 000000011 000000011 100000001 011000011 011110101 111111110	{4}
3246	9	1/4	0.299312	{3, 4, 4, 3}	(1.5708, 0)	-3.14159	000001001 000000111 000000111 000000011 000000011 100000001 011000011 011110101 111111110	{4}
3247	9	1/4	0.292893	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000010 000000010 000000001 000000001 110000001 001100001 110011110	{4}
3248	9	1/4	0.292893	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000010 000000010 000000001 000000001 110000001 001100001 110011110	{4}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3249	9	1/4	0.255479	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100000 111111100	{4}
3250	9	1/4	0.255479	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100000 111111100	{4}
3251	9	1/4	0.228764	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100001 111011110	{4}
3252	9	1/4	0.228764	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000010 000000001 000000001 110000001 001100001 111011110	{4}
3253	9	1/4	0.171573	{4, 5, 4, 5}	(0, 0)	0	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100001 111111110	{4}
3254	9	1/4	0.171573	{4, 5, 5, 4}	(1.5708, 0)	-3.14159	000000101 000000101 000000011 000000011 000000001 000000001 110000001 001100001 111111110	{4}
3255	9	1/5	83.0132	{6, 7, 6, 7}	(0, 0)	0	000100111 000011111 000011111 100000001 011000110 011000110 111011001 111011001 111100110	{4}
3256	9	1/5	83.0132	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000100111 000011111 000011111 100000001 011000110 011000110 111011001 111011001 111100110	{4}
3257	9	1/5	8.23607	{2, 8, 6, 4}	(0, 0)	0	000001100 000001010 000000100 000000011 000000001 110000000 101000001 010100000 000110100	{16}
3258	9	1/5	8.23607	{2, 8, 4, 6}	(1.5708, 0)	3.14159	000001100 000001010 000000100 000000011 000000001 110000000 101000001 010100000 000110100	{16}
3259	9	1/5	75.7771	{7, 8, 7, 8}	(0, 0)	0	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{16}
3260	9	1/5	75.7771	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001011 000001011 000000111 000000111 000000100 110000000 001110000 111100000 111100000	{16}
3261	9	1/5	7.92705	{2, 3, 2, 3}	(0, 0)	0	000001010 000001001 000000111 000000111 000000100 110000000 001110000 101100001 011100010	{8}
3262	9	1/5	7.92705	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001010 000001001 000000111 000000111 000000100 110000000 001110000 101100001 011100010	{8}
3263	9	1/5	7.63932	{0, 1, 0, 1}	(0, 0)	0	000011011 000011011 000000100 000000100 110000010 110000001 001100011 110010100 110001100	{8}
3264	9	1/5	7.63932	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011011 000011011 000000100 000000100 110000010 110000001 001100011 110010100 110001100	{8}
3265	9	1/5	7.54509	{1, 6, 5, 2}	(0, 0)	0	000010010 000001001 000000101 000000011 100000001 010000001 001000001 100100000 011111100	{8}
3266	9	1/5	7.54509	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010010 000001001 000000101 000000011 100000001 010000001 001000001 100100000 011111100	{8}
3267	9	1/5	7.04627	{0, 1, 0, 1}	(0, 0)	0	000111111 000111111 000010111 110001111 111000111 110100010 111110001 111111000 111110100	{4}
3268	9	1/5	7.04627	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000111111 000111111 000010111 110001111 111000111 110100010 111110001 111111000 111110100	{4}
3269	9	1/5	63.4164	{6, 7, 6, 7}	(0, 0)	0	000101111 000011111 000010001 100001111 011000111 110100110 110111001 110111001 111110110	{4}
3270	9	1/5	63.4164	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000101111 000011111 000010001 100001111 011000111 110100110 110111001 110111001 111110110	{4}
3271	9	1/5	6.58359	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000000110 110000001 111000001 111100011 111100100 111011100	{16}
3272	9	1/5	6.58359	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000000110 110000001 111000001 111100011 111100100 111011100	{16}
3273	9	1/5	6	{0, 4, 5, 1}	(0, 0)	0	000010010 000001010 000000110 000000101 100000001 010000001 001100000 111000000 000111000	{8}
3274	9	1/5	6	{0, 4, 1, 5}	(1.5708, 0)	3.14159	000010010 000001010 000000110 000000101 100000001 010000001 001100000 111000000 000111000	{8}
3275	9	1/5	57.8885	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000011011 000000101 111000011 111000011 110100001 111011000 111111100	{4}
3276	9	1/5	57.8885	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000011011 000000101 111000011 111000011 110100001 111011000 111111100	{4}
3277	9	1/5	56.8328	{7, 8, 7, 8}	(0, 0)	0	000101011 000010111 000001111 100000000 010000111 101000000 011010000 111010000 111010000	{4}
3278	9	1/5	56.8328	{7, 8, 8, 7}	(1.5708, 0)	-3.14159	000101011 000010111 000001111 100000000 010000111 101000000 011010000 111010000 111010000	{4}
3279	9	1/5	53.3607	{0, 4, 3, 1}	(0, 0)	0	000101111 000011111 000000011 100001111 010001111 110110100 110111000 111110001 111110010	{8}
3280	9	1/5	53.3607	{0, 4, 1, 3}	(1.5708, 0)	3.14159	000101111 000011111 000000011 100001111 010001111 110110100 110111000 111110001 111110010	{8}
3281	9	1/5	5.62868	{6, 7, 6, 7}	(0, 0)	0	000110111 000101001 000011111 110001111 101000110 011100111 101111001 101111001 111101110	{4}
3282	9	1/5	5.62868	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	000110111 000101001 000011111 110001111 101000110 011100111 101111001 101111001 111101110	{4}
3283	9	1/5	45.1246	{2, 3, 2, 3}	(0, 0)	0	000010100 000001011 000000110 000000110 100000001 010000011 101100001 011101000 010011100	{16}
3284	9	1/5	45.1246	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010100 000001011 000000110 000000110 100000001 010000011 101100001 011101000 010011100	{16}
3285	9	1/5	4.87539	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{8}
3286	9	1/5	4.87539	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000001100 100000001 011100011 011100001 111001001 111011110	{8}
3287	9	1/5	4.81966	{1, 6, 5, 2}	(0, 0)	0	000001101 000001000 000000100 000000011 000000011 110000001 101000001 000110000 100111100	{32}
3288	9	1/5	4.81966	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000001101 000001000 000000100 000000011 000000011 110000001 101000001 000110000 100111100	{32}
3289	9	1/5	4.62617	{7, 8, 7, 8}	(0, 0)	0	000101111 000011111 000011111 100000011 011001100 111010011 111010011 111101100 111101100	{4}
3290	9	1/5	4.62617	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000101111 000011111 000011111 100000011 011001100 111010011 111010011 111101100 111101100	{4}
3291	9	1/5	4.34164	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001010 000000111 000000111 111000000 110110001 111110000 110110100	{4}
3292	9	1/5	4.34164	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001010 000000111 000000111 111000000 110110001 111110000 110110100	{4}
3293	9	1/5	4	{1, 2, 1, 2}	(0, 0)	0	000010001 000001111 000001111 000000011 100000001 011000101 011001001 011100001 111111110	{12}
3294	9	1/5	4	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010001 000001111 000001111 000000011 100000001 011000101 011001001 011100001 111111110	{12}
3295	9	1/5	38.5795	{6, 7, 6, 7}	(0, 0)	0	001011110 000110111 100011000 010001111 111000111 101100111 110111001 110111001 010111110	{4}
3296	9	1/5	38.5795	{6, 7, 7, 6}	(1.5708, 0)	-3.14159	001011110 000110111 100011000 010001111 111000111 101100111 110111001 110111001 010111110	{4}
3297	9	1/5	3.96556	{0, 4, 5, 1}	(0, 0)	0	000010100 000001100 000000101 000000011 100000011 010000011 111000001 000111000 001111100	{16}
3298	9	1/5	3.96556	{0, 4, 1, 5}	(1.5708, 0)	3.14159	000010100 000001100 000000101 000000011 100000011 010000011 111000001 000111000 001111100	{16}
3299	9	1/5	3.63932	{2, 6, 7, 3}	(0, 0)	0	000011111 000001110 000000101 000000011 100000001 110000111 111001000 110101000 101111000	{8}
3300	9	1/5	3.63932	{2, 6, 3, 7}	(1.5708, 0)	3.14159	000011111 000001110 000000101 000000011 100000001 110000111 111001000 110101000 101111000	{8}
3301	9	1/5	3.5	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000010 100000000 010000001 001000001 100100001 111001110	{8}
3302	9	1/5	3.5	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000010 100000000 010000001 001000001 100100001 111001110	{8}
3303	9	1/5	3.16718	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000001011 000000100 000000011 111000000 110100000 111010001 111010010	{4}
3304	9	1/5	3.16718	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000001011 000000100 000000011 111000000 110100000 111010001 111010010	{4}
3305	9	1/5	3.02786	{0, 1, 0, 1}	(0, 0)	0	000001011 000001011 000000110 000000101 000000011 110000000 001100000 111010001 110110010	{8}
3306	9	1/5	3.02786	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001011 000001011 000000110 000000101 000000011 110000000 001100000 111010001 110110010	{8}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3307	9	1/5	28.9443	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000101 100000010 111000011 111100001 111011000 111101100	{8}
3308	9	1/5	28.9443	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000101 100000010 111000011 111100001 111011000 111101100	{8}
3309	9	1/5	27.1803	{1, 6, 5, 2}	(0, 0)	0	000010010 000001001 000000101 000000011 100000000 010000001 001000001 100100000 011101100	{16}
3310	9	1/5	27.1803	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010010 000001001 000000101 000000011 100000000 010000001 001000001 100100000 011101100	{16}
3311	9	1/5	22.1115	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000000101 000000100 110000011 110000011 111100001 110011000 111011100	{4}
3312	9	1/5	22.1115	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000000101 000000100 110000011 110000011 111100001 110011000 111011100	{4}
3313	9	1/5	21.7082	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000101 000000010 000000010 110000000 111000001 110110000 111000100	{8}
3314	9	1/5	21.7082	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000101 000000010 000000010 110000000 111000001 110110000 111000100	{8}
3315	9	1/5	20	{1, 2, 1, 2}	(0, 0)	0	000010001 000001111 000001111 000000110 100000001 011000001 011100011 011100101 111011110	{4}
3316	9	1/5	20	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010001 000001111 000001111 000000110 100000001 011000001 011100011 011100101 111011110	{4}
3317	9	1/5	2.90983	{1, 6, 7, 2}	(0, 0)	0	000010111 000001101 000001011 000000110 100000001 011000000 110100000 101100000 111010000	{8}
3318	9	1/5	2.90983	{1, 6, 2, 7}	(1.5708, 0)	3.14159	000010111 000001101 000001011 000000110 100000001 011000000 110100000 101100000 111010000	{8}
3319	9	1/5	2.80902	{1, 6, 5, 2}	(0, 0)	0	000010010 000001001 000000101 000000011 100000001 010000000 001000000 100100000 011110000	{8}
3320	9	1/5	2.80902	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010010 000001001 000000101 000000011 100000001 010000000 001000000 100100000 011110000	{8}
3321	9	1/5	2.51471	{2, 3, 2, 3}	(0, 0)	0	000010101 000001111 000000011 000000011 100000101 010000010 110010010 011101100 111110000	{8}
3322	9	1/5	2.51471	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000010101 000001111 000000011 000000011 100000101 010000010 110010010 011101100 111110000	{8}
3323	9	1/5	2.45898	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100001 111101110	{8}
3324	9	1/5	2.45898	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000011 100000000 010000001 001000001 100100001 111101110	{8}
3325	9	1/5	2.45683	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000001111 000001101 110000010 111100011 111100011 111011100 111101100	{4}
3326	9	1/5	2.45683	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001101 110000010 111100011 111100011 111011100 111101100	{4}
3327	9	1/5	2.33953	{2, 3, 2, 3}	(0, 0)	0	000011011 000010100 000001111 000001111 110000110 101100111 011111001 101111001 101101110	{4}
3328	9	1/5	2.33953	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000010100 000001111 000001111 110000110 101100111 011111001 101111001 101101110	{4}
3329	9	1/5	2.22222	{1, 2, 1, 2}	(0, 0)	0	000011110 000001111 000001111 000000011 100001110 111010101 111011001 111110001 011101110	{4}
3330	9	1/5	2.22222	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011110 000001111 000001111 000000011 100001110 111010101 111011001 111110001 011101110	{4}
3331	9	1/5	2.1459	{0, 5, 6, 1}	(0, 0)	0	000011010 000010110 000001111 000000011 110000011 101000001 011000001 111110001 001111110	{8}
3332	9	1/5	2.1459	{0, 5, 1, 6}	(1.5708, 0)	3.14159	000011010 000010110 000001111 000000011 110000011 101000001 011000001 111110001 001111110	{8}
3333	9	1/5	2.05573	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{16}
3334	9	1/5	2.05573	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100000 111100000	{16}
3335	9	1/5	15.4721	{2, 8, 7, 3}	(0, 0)	0	000010111 000001011 000000010 000000001 100000111 010000011 100010000 111011000 110111000	{8}
3336	9	1/5	15.4721	{2, 8, 3, 7}	(1.5708, 0)	3.14159	000010111 000001011 000000010 000000001 100000111 010000011 100010000 111011000 110111000	{8}
3337	9	1/5	14.8197	{0, 5, 4, 1}	(0, 0)	0	000010011 000001011 000000101 000000101 100000011 010000011 001100000 110011001 111110010	{8}
3338	9	1/5	14.8197	{0, 5, 1, 4}	(1.5708, 0)	3.14159	000010011 000001011 000000101 000000101 100000011 010000011 001100000 110011001 111110010	{8}
3339	9	1/5	118.138	{0, 1, 0, 1}	(0, 0)	0	000011111 000011111 000010111 000001010 111000101 110100111 111011001 111101000 111011100	{4}
3340	9	1/5	118.138	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000011111 000011111 000010111 000001010 111000101 110100111 111011001 111101000 111011100	{4}
3341	9	1/5	11.5777	{7, 8, 7, 8}	(0, 0)	0	000100111 000011111 000011111 100000000 011001011 011010000 111000011 111010100 111010100	{4}
3342	9	1/5	11.5777	{7, 8, 8, 7}	(1.5708, 0)	-3.14159	000100111 000011111 000011111 100000000 011001011 011010000 111000011 111010100 111010100	{4}
3343	9	1/5	11.0557	{7, 8, 7, 8}	(0, 0)	0	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{12}
3344	9	1/5	11.0557	{7, 8, 8, 7}	(1.5708, 0)	3.14159	000001100 000000111 000000111 000000011 000000011 100000000 111000000 011110000 011110000	{12}
3345	9	1/5	1.96215	{0, 1, 4, 3}	(1.5708, 0)	1.40974	000001100 000001010 000000110 000000101 000000011 110000001 101100000 011010000 000111000	{4}
3346	9	1/5	1.96215	{0, 1, 3, 4}	(1.5708, 0)	-1.73185	000001100 000001010 000000110 000000101 000000011 110000001 101100000 011010000 000111000	{4}
3347	9	1/5	1.95492	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000001 010000001 001000001 100100001 111111110	{8}
3348	9	1/5	1.95492	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000011 100000001 010000001 001000001 100100001 111111110	{8}
3349	9	1/5	1.86223	{1, 2, 1, 2}	(0, 0)	0	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
3350	9	1/5	1.86223	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010011 000001111 000001111 000000011 100000001 011000011 011000001 111101001 111111110	{8}
3351	9	1/5	1.76393	{1, 6, 5, 3}	(0, 0)	0	000001011 000001000 000000111 000000100 000000011 110000110 001101001 101011001 101010110	{16}
3352	9	1/5	1.76393	{1, 6, 3, 5}	(1.5708, 0)	3.14159	000001011 000001000 000000111 000000100 000000011 110000110 001101001 101011001 101010110	{16}
3353	9	1/5	1.69098	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000010 100000000 010000000 001000000 100100001 111000010	{8}
3354	9	1/5	1.69098	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000010 100000000 010000000 001000000 100100001 111000010	{8}
3355	9	1/5	1.61301	{2, 3, 2, 3}	(0, 0)	0	000011110 000010011 000001111 000001111 110000001 101100011 101100001 111101001 011111110	{8}
3356	9	1/5	1.61301	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011110 000010011 000001111 000001111 110000001 101100011 101100001 111101001 011111110	{8}
3357	9	1/5	1.55279	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100000 111110000	{8}
3358	9	1/5	1.55279	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100000 111110000	{8}
3359	9	1/5	1.52786	{2, 3, 2, 3}	(0, 0)	0	000011010 000010101 000001111 000001111 110000011 101100010 011100001 101111000 011110100	{20}
3360	9	1/5	1.52786	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011010 000010101 000001111 000001111 110000011 101100010 011100001 101111000 011110100	{20}
3361	9	1/5	1.40325	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100001 111100010	{8}
3362	9	1/5	1.40325	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000011 100000000 010000000 001000000 100100001 111100010	{8}
3363	9	1/5	1.35721	{1, 2, 1, 2}	(0, 0)	0	000011111 000001111 000001111 000000011 100000011 111000110 111001001 111111001 111110110	{12}
3364	9	1/5	1.35721	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000011111 000001111 000001111 000000011 100000011 111000110 111001001 111111001 111110110	{12}

Table A.1: Parameters for all widgets identified in Chapter 4. (cont.)

ID	N	k/π	ℓ	Attach.	(θ, ϕ)	α	Adjacency matrix	Equivalent
3365	9	1/5	1.26393	{1, 6, 5, 2}	(0, 0)	0	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100001 111110010	{8}
3366	9	1/5	1.26393	{1, 6, 2, 5}	(1.5708, 0)	3.14159	000010011 000001001 000000101 000000011 100000001 010000000 001000000 100100001 111110010	{8}
3367	9	1/5	1.20976	{0, 1, 0, 1}	(0, 0)	0	000001111 000001111 000000111 000000111 000000011 110000000 111100000 111110001 111110010	{4}
3368	9	1/5	1.20976	{0, 1, 1, 0}	(1.5708, 0)	-3.14159	000001111 000001111 000000111 000000111 000000011 110000000 111100000 111110001 111110010	{4}
3369	9	1/5	1.1734	{2, 3, 2, 3}	(0, 0)	0	000011011 000010100 000001111 000001111 110000011 101100011 011100011 101111101 101111110	{4}
3370	9	1/5	1.1734	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011011 000010100 000001111 000001111 110000011 101100011 011100011 101111101 101111110	{4}
3371	9	1/5	1.15654	{2, 3, 2, 3}	(0, 0)	0	000001100 000001001 000000111 000000111 000000010 110000111 101101001 001111000 011101100	{8}
3372	9	1/5	1.15654	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001100 000001001 000000111 000000111 000000010 110000111 101101001 001111000 011101100	{8}
3373	9	1/5	1.11456	{1, 2, 1, 2}	(0, 0)	0	000010111 000001111 000001111 000000011 100000111 011000111 111011000 111111001 111111010	{4}
3374	9	1/5	1.11456	{1, 2, 2, 1}	(1.5708, 0)	-3.14159	000010111 000001111 000001111 000000011 100000111 011000111 111011000 111111001 111111010	{4}
3375	9	1/5	0.756966	{2, 3, 2, 3}	(0, 0)	0	000001010 000001001 000000111 000000111 000000011 110000111 001101000 101111001 011111010	{8}
3376	9	1/5	0.756966	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000001010 000001001 000000111 000000111 000000011 110000111 001101000 101111001 011111010	{8}
3377	9	1/5	0.633667	{2, 3, 2, 3}	(0, 0)	0	000011111 000011111 000001111 000001111 110000011 111100010 111100001 111111000 111110100	{8}
3378	9	1/5	0.633667	{2, 3, 3, 2}	(1.5708, 0)	-3.14159	000011111 000011111 000001111 000001111 110000011 111100010 111100001 111111000 111110100	{8}
3379	9	1/5	0.616115	{6, 7, 6, 7}	(0, 0)	0	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{8}
3380	9	1/5	0.616115	{6, 7, 7, 6}	(1.5708, 0)	3.14159	000011111 000011111 000001111 000001111 110000001 111100001 111100000 111100000 111111000	{8}

Table A.1: Listing of graph parameters for all single-qubit widgets identified in Chapter 4.

References

[1] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I", Monatshefte für Mathematik **38**, 173 (1931); E. Nagel, J. Newman, and D. Hofstadter, *Gödel's Proof* (New York University Press, 2001).

[2] A. Church, "An unsolvable problem of elementary number theory", American Journal of Mathematics **58**, 345 (1936).

[3] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", Proceedings of the London Mathematical Society **s2-42**, 230 (1937).

[4] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge Series on Information and the Natural Sciences (Cambridge University Press, 2000).

[5] I. Bárány and Z. Füredi, "Computing the volume is difficult", Discrete & Computational Geometry **2**, 319 (1987).

[6] M. Dyer, A. Frieze, and R. Kannan, "A random polynomial-time algorithm for approximating the volume of convex bodies", Journal of the ACM **38**, 1 (1991).

[7] R. Feynman, "Simulating physics with computers", International Journal of Theoretical Physics **21**, 467 (1982).

[8] TOP500 Supercomputer Sites, *Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom*, 2012.

[9] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer", Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **400**, 97 (1985).

[10] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation", Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **439**, 553 (1992).

[11] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", in Proceedings of the 35th Annual Symposium on Foundations of Computer Science (1994), pp. 124–134.

[12] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM **21**, 120 (1978).

[13] L. Grover, "A fast quantum mechanical algorithm for database search", in STOC '96, Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (ACM, 1996), pp. 212–219, arXiv:quant-ph/9605043.

[14] W. Wootters and W. Zurek, "A single quantum cannot be cloned", Nature **299**, 802 (1982).

[15] P. Høyer and R. Špalek, "Quantum circuits with unbounded fan-out", in *STACS 2003*, Vol. 2607, edited by H. Alt and M. Habib, Lecture Notes in Computer Science (Springer Berlin / Heidelberg, 2003), pp. 234–246; P. Høyer and R. Špalek, "Quantum fan-out is powerful", Theory of Computing **1**, 83, arXiv:quant-ph/0208043 (2005).

[16] C. Bennett, "Logical reversibility of computation", IBM Journal of Research and Development **17**, 525 (1973).

[17] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm", Quantum Information and Computation **6**, 81, arXiv:quant-ph/0505030 (2006).

[18] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, 2007).

[19] J. Zhang, J. Vala, S. Sastry, and K. B. Whaley, "Exact two-qubit universal quantum circuit", Physical Review Letters **91**, 027903, arXiv:quant-ph/0212109 (2003).

[20] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation", in Quantum Information Science and Its Contributions to Mathematics, Proceedings of Symposia in Applied Mathematics, Vol. 68 (2009), p. 13, arXiv:0904.2557 [quant-ph].

[21] N. Mermin, *Quantum Computer Science: An Introduction* (Cambridge University Press, 2007).

[22] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory", Physical Review A **52**, R2493 (1995).

[23] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error", in STOC '97, Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (1997), pp. 176–188, arXiv:quant-ph/9906129.

[24] A. M. Steane, "Error correcting codes in quantum theory", Physical Review Letters **77**, 793 (1996).

[25] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist", Physical Review A **54**, 1098, arXiv:quant-ph/9512032 (1996).

[26] A. Steane, "Multiple-particle interference and quantum error correction", Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **452**, 2551, arXiv:quant-ph/9601029 (1996).

[27] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets", Physical Review Letters **102**, 110502, arXiv:0811.4262 [quant-ph] (2009).

[28] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states", Physical Review A **68**, 022312, arXiv:quant-ph/0301052 (2003).

[29] D. Gross and J. Eisert, "Novel schemes for measurement-based quantum computation", Physical Review Letters **98**, 220503, arXiv:quant-ph/0609149 (2007).

[30] A. G. D'Souza and D. L. Feder, "Strategies for measurement-based quantum computation with cluster states transformed by stochastic local operations and classical communication", Physical Review A **84**, 042301, arXiv:1108.4909 [quant-ph] (2011).

[31] N. Biggs, *Algebraic Graph Theory* (Cambridge University Press, 1974).

[32] C. Godsil and G. Royle, *Algebraic Graph Theory*, Graduate Texts in Mathematics (Springer, 2001).

[33] G. Agnarsson and R. Greenlaw, *Graph Theory: Modeling, Applications, and Algorithms*, Pearson international edition (Pearson/Prentice Hall, 2007).

[34] Y. Aharonov, L. Davidovich, and N. Zagury, "Quantum random walks", Physical Review A **48**, 1687 (1993).

[35] J. Kempe, "Quantum random walks: an introductory overview", Contemporary Physics **44**, 307, arXiv:quant-ph/0303081 (2003).

[36] S. Venegas-Andraca, "Quantum walks: A comprehensive review", Quantum Information Processing **11**, 1015, arXiv:1201.4780 [quant-ph] (2012).

[37] E. Farhi and S. Gutmann, "Quantum computation and decision trees", Physical Review A **58**, 915, arXiv:quant-ph/9706062 (1998).

[38] F. W. Strauch, "Connecting the discrete- and continuous-time quantum walks", Physical Review A **74**, 030301, arXiv:quant-ph/0606050 (2006).

[39] A. Childs, "On the relationship between continuous- and discrete-time quantum walk", Communications in Mathematical Physics **294**, 581, arXiv:0810.0312 [quant-ph] (2010).

[40] S. Bose, "Quantum communication through an unmodulated spin chain", Physical Review Letters **91**, 207901, arXiv:quant-ph/0212041 (2003).

[41] M. Christandl, N. Datta, A. Ekert, and A. J. Landahl, "Perfect state transfer in quantum spin networks", Physical Review Letters **92**, 187902, arXiv:quant-ph/0309131 (2004).

[42] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, and A. J. Landahl, "Perfect transfer of arbitrary states in quantum spin networks", Physical Review A **71**, 032312, arXiv:quant-ph/0411020 (2005).

[43] S. Bose, "Quantum communication through spin chain dynamics: an introductory overview", Contemporary Physics **48**, 13, arXiv:0802.1224 [cond-mat.other] (2007).

[44] C. Godsil, "When can perfect state transfer occur?", arXiv:1011.0231 [math.CO] (2010).

[45] C. Godsil, "Periodic graphs", The Electronic Journal of Combinatorics **18**, P23, arXiv:0806.2074 [math.CO] (2011).

[46] A. Fetter and A. Walecka, *Quantum Theory of Many-Particle Systems*, Dover Books on Physics (Dover Publications, 1971).

[47] A. Ambainis, "Quantum walks and their algorithmic applications", International Journal of Quantum Information **01**, 507, arXiv:quant-ph/0403120 (2003).

[48] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, "Quantum walks on graphs", in STOC '01, Proceedings of the thirty-third annual ACM symposium on Theory of computing, STOC '01 (2001), pp. 50–59, arXiv:quant-ph/0012090.

[49] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, "One-dimensional quantum walks", in STOC '01, Proceedings of the thirty-third annual ACM symposium on Theory of computing, STOC '01 (2001), pp. 37–49.

[50] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm", Physical Review A **67**, 052307, arXiv:quant-ph/0210064 (2003).

[51] M. Santha, "Quantum walk based search algorithms", in TAMC, Vol. 4978, edited by M. Agrawal, D.-Z. Du, Z. Duan, and A. Li, Lecture Notes in Computer Science (2008), pp. 31–46, arXiv:0808.0059 [quant-ph].

[52] A. Ambainis, "Quantum walk algorithm for element distinctness", in Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on (Oct. 2004), pp. 22 –31; A. Ambainis, "Quantum walk algorithm for element distinctness", SIAM Journal on Computing **37**, 210, arXiv:quant-ph/0311001 (2007).

[53] A. M. Childs and J. M. Eisenberg, "Quantum algorithms for subset finding", Quantum Information and Computation **5**, 593, arXiv:quant-ph/0311038 (2005).

[54] F. Magniez, M. Santha, and M. Szegedy, "Quantum algorithms for the triangle problem", in Symposium on Discrete Algorithms: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Vol. 23 (2005), pp. 1109–1117; F. Magniez, M. Santha, and M. Szegedy, "Quantum algorithms for the triangle problem", SIAM Journal on Computing **37**, 413, arXiv:quant-ph/0310134 (2007).

[55] S. Cook, "The complexity of theorem-proving procedures", in STOC '71, Proceedings of the third annual ACM symposium on Theory of computing (ACM, 1971), pp. 151–158.

[56] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by a quantum walk", in STOC '03, Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (2003), pp. 59–68, arXiv:quant-ph/0209131.

[57] A. M. Childs, E. Farhi, and S. Gutmann, "An example of the difference between quantum and classical random walks", Quantum Information Processing **1**, 35, arXiv:quant-ph/0103020 (2002).

[58] R. Cleve, D. Gottesman, M. Mosca, R. D. Somma, and D. Yonge-Mallo, "Efficient discrete-time simulations of continuous-time quantum query algorithms", in STOC '09, Proceedings of the 41st annual ACM symposium on Theory of computing (2009), pp. 409–416, arXiv:0811.4428 [quant-ph].

[59] D. W. Berry, R. Cleve, and S. Gharibian, "Gate-efficient discrete simulations of continuous-time quantum query algorithms", arXiv:1211.4637 [quant-ph] (2012).

[60] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum algorithm for the Hamiltonian NAND tree", Theory of Computing **4**, 169, arXiv:quant-ph/0702144 (2008).

[61] M. Varbanov and T. A. Brun, "Quantum scattering theory on graphs with tails", Physical Review A **80**, 052330, arXiv:0906.2825 [quant-ph] (2009).

[62] A. M. Childs, "Universal computation by quantum walk", Physical Review Letters **102**, 180501, arXiv:0806.1972 [quant-ph] (2009).

[63] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon, "Universal quantum computation using the discrete-time quantum walk", Physical Review A **81**, 042330, arXiv:0910.1024 [quant-ph] (2010).

[64] B. A. Blumer, M. S. Underwood, and D. L. Feder, "Single-qubit unitary gates by graph scattering", Physical Review A **84**, 062302, arXiv:1111.5032 [quant-ph] (2011).

[65] The Online Encyclopedia of Integer Sequences, *Number of graphs on n unlabeled nodes*, [http://oeis.org/A000088] and references therein, accessed 21 September, 2012.

[66] B. D. McKay, *The nauty page*, [http://cs.anu.edu.au/∼bdm/nauty], accessed 21 September, 2012.

[67] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi, *GNU Scientific Library Reference Manual (3rd Ed.)* (Network Theory, Jan. 2009); Free Software Foundation, Inc., *GSL—GNU Scientific Library—GNU Project—Free Software Foundation (FSF)*, [http://www.gnu.org/software/gsl/], accessed 21 September, 2012.

[68] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide (3rd Ed.)* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999); Netlib, *LAPACK—Linear Algebra PACKage*, [http://netlib.org/lapack/], accessed 21 September, 2012.

[69] A. M. Childs, D. Gosset, and Z. Webb, "Universal computation by multi-particle quantum walk", arXiv:1205.3782 [quant-ph] (2012).

[70] J. Du, H. Li, X. Xu, M. Shi, J. Wu, X. Zhou, and R. Han, "Experimental implementation of the quantum random-walk algorithm", Physical Review A **67**, 042316, arXiv:quant-ph/0203120 (2003).

[71] C. A. Ryan, M. Laforest, J. C. Boileau, and R. Laflamme, "Experimental implementation of a discrete-time quantum random walk on an NMR quantum-information processor", Physical Review A **72**, 062317, arXiv:quant-ph/0507267 (2005).

[72] H. B. Perets, Y. Lahini, F. Pozzi, M. Sorel, R. Morandotti, and Y. Silberberg, "Realization of quantum walks with negligible decoherence in waveguide lattices", Physical Review Letters **100**, 170506, arXiv:0707.0741 [quant-ph] (2008).

[73] M. Karski, L. Förster, J.-M. Choi, A. Steffen, W. Alt, D. Meschede, and A. Widera, "Quantum walk in position space with single optically trapped atoms", Science **325**, 174, arXiv:0907.1565 [quant-ph] (2009).

[74] A. Peruzzo, M. Lobino, J. C. F. Matthews, N. Matsuda, A. Politi, K. Poulios, X.-Q. Zhou, Y. Lahini, N. Ismail, K. Wörhoff, Y. Bromberg, Y. Silberberg, M. G. Thompson, and J. L. O'Brien, "Quantum walks of correlated photons", Science **329**, 1500, arXiv:1006.4764 [quant-ph] (2010).

[75] A. Schreiber, K. N. Cassemiro, V. Potoček, A. Gábris, P. J. Mosley, E. Andersson, I. Jex, and Ch. Silberhorn, "Photons walking the line: a quantum walk with adjustable coin operations", Physical Review Letters **104**, 050502, arXiv:0910.2197 [quant-ph] (2010).

[76] F. Zähringer, G. Kirchmair, R. Gerritsma, E. Solano, R. Blatt, and C. F. Roos, "Realization of a quantum walk with one and two trapped ions", Physical Review Letters **104**, 100503, arXiv:0911.1876 [quant-ph] (2010).

[77] M. S. Underwood and D. L. Feder, "Universal quantum computation by discontinuous quantum walk", Physical Review A **82**, 042304, arXiv:1008.3578 [quant-ph] (2010).

[78] P. P. Rohde, A. Schreiber, M. Štefaňák, I. Jex, and C. Silberhorn, "Multi-walker discrete time quantum walks on arbitrary graphs, their properties and their photonic implementation", New Journal of Physics **13**, 013001, arXiv:1006.5556 [quant-ph] (2011).

[79] P. Xue and B. C. Sanders, "Two quantum walkers sharing coins", Physical Review A **85**, 022307, arXiv:1112.1487 [quant-ph] (2012).

[80] I. Carneiro, M. Loo, X. Xu, M. Girerd, V. Kendon, and P. L. Knight, "Entanglement in coined quantum walks on regular graphs", New Journal of Physics **7**, 156, arXiv:quant-ph/0504042 (2005).

[81] J. K. Gamble, M. Friesen, D. Zhou, R. Joynt, and S. N. Coppersmith, "Two-particle quantum walks applied to the graph isomorphism problem", Physical Review A **81**, 052313 (2010).

[82] J. Smith, "On the limitations of graph invariants inspired by quantum walks", Electronic Notes in Discrete Mathematics **38**, The Sixth European Conference on Combinatorics, Graph Theory and Applications, EuroComb 2011, 795 (2011).

[83] K. Rudinger, J. K. Gamble, E. Bach, M. Friesen, R. Joynt, and S. N. Coppersmith, "Comparing algorithms for graph isomorphism using discrete-and continuous-time quantum random walks", Arxiv preprint arXiv:1207.4535 (2012).

[84] M. S. Underwood and D. L. Feder, "Bose–Hubbard model for universal quantum-walk-based computation", Physical Review A **85**, 052314, arXiv:1204.6021 [quant-ph] (2012).

[85] M. P. A. Fisher, P. B. Weichman, G. Grinstein, and D. S. Fisher, "Boson localization and the superfluid-insulator transition", Physical Review B **40**, 546 (1989).

[86] D. Jaksch, H.-J. Briegel, J. I. Cirac, C. W. Gardiner, and P. Zoller, "Entanglement of atoms via cold controlled collisions", Physical Review Letters **82**, 1975, arXiv:quant-ph/9810087 (1999).

[87] H. Briegel, T. Calarco, D. Jaksch, J. Cirac, and P. Zoller, "Quantum computing with neutral atoms", Journal of Modern Optics **47**, 415, arXiv:quant-ph/9904010 (2000).

[88] I. Bloch, "Quantum coherence and entanglement with ultracold atoms in optical lattices", Nature **453**, 1016 (2008).

[89] R. Ionicioiu and P. Zanardi, "Quantum-information processing in bosonic lattices", Physical Review A **66**, 50301, arXiv:quant-ph/0204118 (2002).

[90] D. L. Feder, "Perfect quantum state transfer with spinor bosons on weighted graphs", Physical Review Letters **97**, 180502, arXiv:quant-ph/0606065 (2006).

[91] J. Mompart, K. Eckert, W. Ertmer, G. Birkl, and M. Lewenstein, "Quantum computing with spatially delocalized qubits", Physical Review Letters **90**, 147901, arXiv:quant-ph/0209171 (2003).

[92] T. Calarco, E. A. Hinds, D. Jaksch, J. Schmiedmayer, J. I. Cirac, and P. Zoller, "Quantum gates with neutral atoms: controlling collisional interactions in time-dependent traps", Physical Review A **61**, 022304, arXiv:quant-ph/9905013 (2000).

[93] S. Bergamini, B. Darquié, M. Jones, L. Jacubowiez, A. Browaeys, and P. Grangier, "Holographic generation of microtrap arrays for single atoms by use of a programmable phase modulator", Journal of the Optical Society of America B **21**, 1889, arXiv:quant-ph/0402020 (2004).

[94] K. Nelson, X. Li, and D. Weiss, "Imaging single atoms in a three-dimensional array", Nature Physics **3**, 556 (2007).

[95] W. Bakr, J. Gillen, A. Peng, S. Fölling, and M. Greiner, "A quantum gas microscope for detecting single atoms in a Hubbard-regime optical lattice", Nature **462**, 74, arXiv:0908.0174 [cond-mat.quant-gas] (2009).

[96] C. Weitenberg, M. Endres, J. Sherson, M. Cheneau, P. Schauß, T. Fukuhara, I. Bloch, and S. Kuhr, "Single-spin addressing in an atomic Mott insulator", Nature **471**, 319, arXiv:1101.2076 [cond-mat.quant-gas] (2011).

[97] Y. Ge, B. Greenberg, O. Perez, and C. Tamon, "Perfect state transfer, graph products and equitable partitions", International Journal of Quantum Information **09**, 823, arXiv:1009.1340 [quant-ph] (2011).

[98] R. Bachman, E. Fredette, J. Fuller, M. Landry, M. Opperman, C. Tamon, and A. Tollefson, "Perfect state transfer on quotient graphs", Quantum Information and Computation **12**, 293, arXiv:1108.0339 [quant-ph] (2012).

[99] C. Godsil, "State transfer on graphs", Discrete Mathematics **312**, 129, arXiv:1102.4898 [math.CO] (2012).

[100] J. Brown, C. Godsil, D. Mallory, A. Raz, and C. Tamon, "Perfect state transfer on signed graphs", arXiv:1211.0505 [quant-ph] (2012).

[101] A. Brouwer and W. Haemers, *Spectra of Graphs*, Universitext Series (Springer, 2011).

[102] C. Bruder, R. Fazio, and G. Schön, "The Bose–Hubbard model: from Josephson junction arrays to optical lattices", Annalen der Physik **14**, 566 (2005).

[103] I. Bloch, J. Dalibard, and W. Zwerger, "Many-body physics with ultracold gases", Reviews of Modern Physics **80**, 885, arXiv:0704.3011 [cond-mat.other] (2008).

[104] M. A. Cazalilla, R. Citro, T. Giamarchi, E. Orignac, and M. Rigol, "One dimensional bosons: from condensed matter systems to ultracold gases", Reviews of Modern Physics **83**, 1405, arXiv:1101.5337 [cond-mat.str-el] (2011).

[105] D. J. Scalapino, "A common thread: the pairing interaction for unconventional superconductors", Rev. Mod. Phys. **84**, 1383, arXiv:1207.4093 [cond-mat.supr-con] (2012).