Order Number 9207048

Cayley networks: A group-theoretic approach to the design and analysis of parallel networks

Baumslag, Marc, Ph.D.

City University of New York, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

CAYLEY NETWORKS: A GROUP-THEORETIC APPROACH TO THE DESIGN AND ANALYSIS OF PARALLEL NETWORKS

by

Marc Baumslag

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1991

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/19/91	Michael Anshel
Date	Chair of Examining Committee
9/19/91	Stanley Habib
Date	Executive Officer

D. Frank Hsu

Rohit Parikh

Christina M. Zamfirescu

Supervisory Committee

The City University of New York

To my parents, Gilbert and Sybil.

ABSTRACT

CAYLEY NETWORKS: A GROUP-THEORETIC APPROACH TO THE DESIGN AND ANALYSIS OF PARALLEL NETWORKS

by

Marc Baumslag

Adviser: Professor Michael Anshel

A **Cayley network** is defined to be a parallel network that has an underlying topology of a Cayley graph. Many families of parallel networks built in practice and proposed in the literature can be characterized as Cayley networks, and a multitude of structural and computational benefits ensue from their underlying group-theoretic structure. Also, the structural uniformity of Cayley networks can be exploited to attack many problems from within a single framework.

In the first part of this thesis we study Cayley networks with respect to the issues of *communication*, *reliability* and the *mapping problem*. We discuss several schemes for performing *point-to-point routing* in Cayley networks and present a general strategy for finding efficient *permutation routes*. The strategy applies to many familiar networks in the literature, including toroidal mesh networks, hypercube networks, butterfly networks, star networks and a large class of double-ring networks. Often, the routes we derive are the most efficient currently known to exist. On the topic of reliability, we study the *connectivity* of Cayley graphs and derive a general inductive tool for this purpose. Specific applications of this tool include new proofs of results by Godsil [49] and Akers and Krishnamurthy [3]. With regard to the mapping problem, we present several methods

for embedding a Cayley graph in a direct product graph. As a consequence, we describe *work-preserving* emulations of Cayley networks on a family of smaller networks, thereby exposing a processor-time tradeoff.

In the second part of this thesis we prove the following combinatorial results:

- We present an embedding of the N-node deBruijn network into the N-node hypercube network with dilation $\lfloor 2/5 \log N \rfloor$ and congestion 2.
- We describe a *deterministic* algorithm for reconfiguring faulty hypercubes. This is the first deterministic solution to this problem.
- We determine a generating set for *iterated wreath products* and *semidirect products* of cyclic groups that endows the corresponding *directed* Cayley graphs with directed Hamiltonian circuits, verifying a conjecture of Klerlein [68] for these classes of groups.

ACKNOWLEDGMENTS

First and foremost, I would like to acknowledge my adviser, Michael Anshel, for all of his encouragement, assistance and generosity during the development and completion of this document. His insights and experience were invaluable to me. I would also like to thank the members of my committee, D. Frank Hsu, Rohit Parikh and Christina Zamfirescu for all of their comments and useful advice. In particular, I would like to thank Rohit Parikh for pointing out an error in an earlier version of Chapter 7.

I would also like to express my gratitude to Arnold Rosenberg. Professor Rosenberg initially introduced me to the subject of parallel networks and pointed me in the direction of many fruitful problems. A large proportion of the research in this thesis has either been directly or indirectly influenced by his comments and ideas.

In addition to those mentioned above, I would like to thank my father, Gilbert Baumslag, for all of his help over the last five years and Victor Yodaiken for his constant support throughout my thesis. Also deserving credit for their helpful comments and criticisms are Bill Aiello, Fred Annexstein, David Mix Barrington, Gary Bloom, Seth Malitz and Seshaiyen Raghuram. In terms of moral support, I would like to thank Jean Orr, Mary Baumslag and Stephen Ahearn. Finally, thanks to my close friend, Avraham Sokolovsky for getting me on the right track.

Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, the domain of parallel computation has entered a new era of expanded possibilities. Due to the low cost of powerful individual processors it is now practical to build parallel networks consisting of many thousands of processing elements. However, the task of exploiting and managing the capabilities of such machines is still in its infancy. In the theoretical community there has been a state of flux regarding the underlying principles that govern parallel computation, and many fundamental architectural problems, such as how to choose an appropriate topology for a network, reliability, and the support of efficient communication, are, to a large extent, unresolved. One shortcoming of current research on these issues is the lack of unifying principles. The addition of *group theory* as a tool both in design and analysis provides such a unifying force, and a primary aim of this thesis is to explore the application of this tool to several key problems.

In the design of a large-scale parallel network, many important issues should be addressed. One of the most crucial is choosing a topology for the underlying architecture and several factors must be taken into account to balance cost and performance. In particular, the type of computations to be supported by the network will be very influential on the choice of the final topology. If the purpose of the network is to support general computations then we may choose a topology which can efficiently emulate a shared memory architecture (using techniques of Ranade [87] or Valiant and Brebner [105], for example). On the other hand, if the purpose of the network is to execute some fixed class of algorithms then we want the topology to conform to the communication requirements of these algorithms as closely as possible. One very natural class of topologies,

subsuming many "benchmark" parallel networks, is characterized by the fact that they are derived from the structure of a finite group. Besides providing a common "umbrella" for studying parallel networks, group-theoretic structure engenders many structural and computational benefits to the parallel network designer. Other important issues that must be addressed are:

- How do processing elements communicate?
- What provision is made for the event of failure of some of the network's components?
- How can an algorithm designed for a virtual parallel network be converted to run on our physical network?

These questions form the focus of this thesis, and in the first part, we concentrate primarily on *Cayley networks*, *i.e.*, networks whose underlying communication topology is a Cayley graph[1]. The next section summarizes the main results of the thesis.

1.1 Summary of Results

An important goal of this thesis is to develop a formal framework for studying parallel networks and apply it to a variety of problems. To exemplify the basic tools of this framework, we define a new class of networks, called *wreathed networks*, which generalizes the cube-connected cycles network [86], providing a denser family of topologies. (For each pair of positive integers n and k, where k is a divisor of n, there exists a wreathed network of size $n2^k$.)

Our first investigation focuses on the problem of routing in Cayley networks. We suggest some basic schemes for point-to-point routing and analyze the underlying origin of efficient labelling schemes for Cayley networks.

[1]All formal definitions appear in Chapter 2.

Another familiar type of routing problem is the problem of routing *permutations* of a network. We develop a series of results on this topic assumming that we have off-line knowledge of the permutation that we want to route. Our main result describes a general strategy for finding efficient routes and we demonstrate its efficacy on a variety of networks, including mesh networks, hypercube networks, butterfly-derivative networks, star networks, and several other classes of Cayley networks. In almost all cases, the routes are congestion-free and their length is within a small constant factor of the diameter of the network. Due to the inherent symmetry of Cayley networks, we conjecture that all Cayley networks have the ability to support efficient (off-line) permutation routing. In an attempt to substantiate this conjecture, we prove that for every finite group, there exists an "efficient" set of generators that endows the corresponding Cayley graph with the ability to route arbitrary permutations.

The second topic we study is that of fault-tolerance. We examine this issue from three different angles (two of which apply primarily to Cayley networks). Firstly, we extend the general permutation routing scheme discussed above to handle faulty processing elements. This provides, for example, a proof that off-line routing can be performed on the Beneš network and the mesh network, each with a single arbitrary faulty processing element, without *any* time loss in comparison to a fault-free network.

Another viewpoint from which to attack the problem of fault-tolerance relates the fault-tolerance of a network to the connectivity of its underlying graph. The connectivity of Cayley graphs has received a fair amount of previous attention. In particular, Godsil has shown that every Cayley graph equipped with a minimal generating set has connectivity equal to the degree of the graph [49]. Also, Akers and Krishnamurthy have shown that every quasi-minimal Cayley graph in which all the elements of the generating set are of order 2 has connectivity equal to its degree [3]. Our main theorem on connectivity, from which we deduce the above two results as corollaries, is an inductive tool

for analyzing the connectivity of Cayley graphs. The central idea of our approach rests on previously analyzed connectivity properties of arbitrary vertex-transitive graphs. We also discuss the implications of our results for fault-tolerant routing in parallel networks, suggesting a fault-tolerant method for packaging a parallel network whose topology is a quasi-minimal Cayley graph.

In our third study of fault-tolerance, we restrict our attention to the hypercube network, and analyze its ability to reconfigure itself in the presence of faulty vertices. Our result hinges on the existence of a decomposition of the n-dimensional hypercube into disjoint sets of size $\Theta(n)$ that arises from the theory of error-correcting codes. Assuming that there are at least αn live vertices in each of these sets, we deterministically embed a fault-free hypercube in the faulty one with dilation 5, load factor at most $\lceil 6/\alpha \rceil + 1$, and unit dynamic congestion. Our algorithm is the first deterministic solution to this problem. (It was previously known that, assuming vertices fail randomly and uniformily with a fixed probability, $p < 1$, a constant dilation, constant congestion embedding can be efficiently obtained with high probability [58].)

We study two graph embedding problems, the first of which applies exclusively to Cayley networks. We derive three different techniques for efficiently embedding a Cayley graph in the direct product of two smaller graphs. This leads to a uniform method for translating algorithms designed for Cayley networks to *quotient networks* of smaller size. In the second embedding, we focus on the relationship between the deBruijn graph and the hypercube. We present an embedding of the order-n deBruijn graph in the order-n hypercube with dilation at most $\lfloor 2n/5 \rfloor + 2$ and congestion 2. This is currently the best upper bound known for this problem. We also present a generalization of our embedding technique, making it applicable to a wide variety of embedding problems.

The final topic of this thesis concerns the search for Hamiltonian circuits in Cayley graphs. This problem has received a lot of attention in recent years and it has been

conjectured that, in the undirected case, all Cayley graphs have a Hamiltonian cycle. In the directed case, it is known that this is not true. However, Klerlein has conjectured that for every finite group, there exists a minimal generating for which the resultant Cayley digraph has a Hamiltonian circuit. We verify this conjecture for several families of groups, including an arbitrary iterated wreath product of cyclic groups and the semidirect product of two cyclic groups.

In the next section, we present an overview of the central issues essential to the study of parallel networks. These issues form the background and practical motivations for the problems we study in this thesis.

1.2 Design Issues for Parallel Networks – An Overview

As we have already mentioned, several important issues need to be addressed in relation to the design of a parallel network. From our point of view, of foremost importance are questions relating to the network's topology, routing and algorithm design, reliability, the mapping problem and scalability. We briefly discuss each of these issues in turn.

1.2.1 Topology Considerations

In the shared memory model of parallel computation each processor can access any word of a global address space in unit time. This model becomes technologically unrealistic as the number of processors increases. However, the shared memory model can be approximated in hardware (there is a logarithmic penalty in memory access time) by connecting the processors to a set of memory modules via an *interconnection network*. Examples include banyan networks, Beneš networks, omega networks and a multitude of other networks known as *multistage interconnection networks*, many of which have been shown to be topologically equivalent [111]. A disadvantage of this category of network is that a *fixed* amount of time is required for each memory access and thus any locality in memory access patterns displayed by the algorithm cannot be exploited.

A third model for a parallel network, overcoming the memory access problems inherent in multistage interconnection networks, comprises a set of processing elements (PEs), each consisting of a processor and a local memory, connected by a fixed communication network. Henceforth, we are concerned exclusively with networks of this type. Access of data in non-local memory occurs via the exchange of *messages.* A plethora of possible topologies have been proposed in the literature, and often, the proposed topology is justified by assumptions regarding the type of algorithms that the network is destined to support. For example, a tree network is appropriate for the parallel evaluation of associative binary operators, such as addition of integers, whereas a mesh network naturally accomodates the type of computation required in low-level image processing.

The richness of the topology menagerie that exists in the literature is an indication of its central role in the study of parallel computing. The simplest topologies are lines and rings. Variants of ring networks have been proposed, such as double-ring networks and circulants, that reduce the linear diameter of a simple ring. Also, many variants of tree topologies have been proposed, including X-trees, mesh-of-trees and fat trees.

The hypercube topology is one of the most widely studied networks in the literature. This is mainly due to its highly regular structure and the concommitant ease of programmability. However, the number of ports per processor increases logarithmically with the number of processors making it impractical when the number of processors is very large. A large class of so-called hypercube-derivative networks have been proposed to overcome this problem; they have bounded degree and retain the capability of executing many hypercube algorithms efficiently. This class includes the shuffle-exchange, deBruijn, butterfly, cube-connected cycles and product-shuffle networks.

In the study of a network's topology, several graph-theoretic parameters and their inter-relationships play an important role (see, for example, [25,38]). *Diameter* measures worst-case latency for point-to-point routing while *connectivity* measures a network's

vulnerability to processor faults and *expansion factor* relates to the ability of a network to perform fast movement of large sets of data (embodied, for instance, in the sorting problem). Other parameters such as *bisection width* and the size of a *separator*, are important for efficient VLSI layouts of a network. Finally, much effort has been oriented towards constructing networks with specific constraints imposed on one or more of the above parameters.

1.2.2 Algorithm Design and Communication

The foremost vehicle for the design of parallel algorithms is the Parallel Random Access Machine (PRAM) in which a set of autonomous processing elements are assumed to be able to access any word of a global address space in unit time. Many efficient parallel algorithms have been designed for this model [43], although, in general, the emulation of a PRAM algorithm on a bounded degree network is accompanied by a slowdown factor caused by communication latency between distant processors in the network. If the communication requirements of an algorithm are completely unknown or do not follow any regular patterns then the PRAM model coupled with an efficient emulation strategy for some fixed network may be quite practical. However, given a priori knowledge about the type of communication patterns to be performed during execution of the algorithm, we may be able to take advantage of the topology of the network. Exploiting this *locality* displayed by the communication pattern of an algorithm, requires an efficient means of interprocessor communication.

The first and most basic type of communication problem that arises is the *point-to-point routing problem*, *i.e.*, the problem of determining efficient paths between pairs of PEs in a network. A simple and efficient point-to-point routing algorithm is useful in that it simplifies the task of programming. This partially explains the success of topologies such as the hypercube and the butterfly in the parallel community. The existence of an efficient point-to-point routing scheme is closely related to finding an appropriate

labelling of the processing elements and links of the network.

A second common form of communication occurs when the movement of data between processors reflects some type of global pattern. If each processor is the origin and destination of a single data item then we have the problem of *permutation routing.* Many important computational problems reduce (or are closely related) to the permutation routing problem, including problems in linear algebra, sorting and the emulation of one network by another.

1.2.3 Approaches to Reliability

Due to imperfections in the fabrication process of computer chips and data transmission media, the chances that some of a network's components are inoperable or will become inoperable at some point during the network's lifetime are very significant. Unless some mechanism is provided for dealing with such faults, the performance of the network may be seriously impaired. From a practical perspective, methods for fault detection and recovery must be built into the network. On the other hand, there are *physical* limitations on the number of faulty components a network can tolerate before it becomes non-functional. The *connectivity* of a network provides a worst-case measure for this in the sense that it measures the minimum number of PEs that must fail to prevent two sets of non-faulty PEs from communicating.

Two major approaches can be taken towards overcoming faulty components in a network. The first incorporates additional *hardware* in the system to compensate for faulty components. An example of this occurs in the DIOGENES design methodology [92] developed to construct fault-tolerant implementations of VLSI processor arrays. In this scheme, enough processors are provided to allow for expected failure. The methodology then consists of translating an abstract *book embedding* of the target network into a physical implementation using the fault-free processors.

A second approach is to provide a recovery mechanism in the *software* of the system. This approach is illustrated by techniques developed by Hastad et al [58] for *reconfiguring* a hypercube network in the presence of faults so that the original machine may *simulate* a virtual fault-free version with only a small penalty in performance. Choosing the type of fault-tolerance to provide in a network involves a tradeoff between cost and performance.

1.2.4 The Mapping Problem for Parallel Networks

Due to the abundance in types of parallel networks, the portability of a parallel program from one machine to another presents a serious problem. One aspect of this problem directs attention to mapping the *communication structure* of a parallel program designed for one network to a different network. A common approach to solving this problem is embodied in the notion of a *graph embedding*. Informally, an embedding of a guest graph in a host graph measures the ability of the host to simulate general computations of the guest. There have been numerous investigations of relationships between different classes of networks along these lines. In particular, the relationship between meshes, hypercubes and hypercube-derivative networks has been the focus of a lot of attention [69,26,36,97]. An outstanding unsolved problem is to determine the comparative power of deBruijn networks and hypercubes. This thesis takes a first step towards a solution to this problem.

1.2.5 Scalability

A network is *scalable* if it can easily be extended to a larger number of processors. This is an important practical problem. It is necessary to be able to increase the size of a network while retaining its vital properties. For example, it is easy to construct a larger mesh network from two smaller mesh networks, assuming the potential for adding communication links to the appropriate existing proceesors has been provided. We would like the addition of new processors to a network to take place in a uniform way, thus

simplifying the production and integration of components. Other important features that should be scalable are routing algorithms, fault-tolerance and structural parameters. Clearly, for a network to be scalable, appropriate features must be built-in to its design.

1.3 The Group-Theoretic Point of View

In order to tackle some of the issues discussed in the previous section, it would be advantageous to have a unified framework in which to work. Although some characteristics of a given network may be peculiar to only that network and hence must be studied with a narrow persepective, many networks share important properties, such as the ability to efficiently route permutations or the resilience to the occurence of faults. A group-theoretic framework enables us to explore many issues for large classes of networks with such a wide perspective. In this section, we examine the reasons for this and review previous work along similar lines.

1.3.1 Why Group Theory?

There is a natural way to form a graph from a group and a set of generators for the group. The vertices of the graph are the elements of the group and there is an edge between two elements when some generator "leads" us from one to the other via multiplication in the group. Graphs constructed in this fashion are called Cayley graphs and comprise an extremely rich class due to the abundance of groups and generating sets that exist. If one models the communication graph of a parallel network by a Cayley graph then it becomes possible to employ group-theoretic techniques towards the analysis of structural and algorithmic properties of the network. This mode of attack is encouraged by the fact that many parallel networks developed both commercially and theoretically have an underlying topology of a Cayley graph such as cube-connected cycles networks, butterfly networks, toroidal mesh networks, hypercube networks and double ring networks. We identify the following additional advantages.

1. Additional level of structure. The standard approach to studying both structural and computational properties of parallel networks is purely combinatorial and graph-theoretic. Besides supplying a new set of tools, the use of group-theoretic structure can expose structural and computational properties of networks in a cleaner and often simpler way compared to corresponding combinatorial approaches.

2. Descriptive power. To describe a Cayley network, we need to specify the underlying group and the appropriate generating set for the group. This may be done symbolically or through familiar methods of group representation such as via a set of permutations or a set of generators and defining relations. In particular, compact representations of groups may be employed for manipulating very large network topologies. This problem arises in the study of distributed networks in which each node retains a copy of the network's structure. Also, an unexplored area in which we believe the descriptive power of Cayley networks should be of value is in their visual representation – an important software tool for algorithm design in a parallel programming environment.

3. Generality. A consequence of approaching problems through a single framework is that results pertaining to large classes of networks can be obtained, allowing us to exploit commonalities among topologies rather than "reinvent the wheel" for each individual topology. For example, in this thesis we show that a large class of Cayley networks have optimal connectivity, including several popular networks such as hypercube networks, star networks and butterfly networks. Clearly, this approach is more economical than studying each network individually.

4. Constructions involving Cayley graphs. Cayley graphs provide a wellspring of possible topologies for important graph-theoretic constructions with applications in the design of parallel networks. These include constructions of *extremal graphs* [12,33] and *expander graphs* [6,74,76].

5. Algorithmic benefits. Several immediate algorithmic benefits can be reaped from

the group structure. In any parallel system an addressing scheme for the PEs and communication links plays a crucial role in algorithm design. The group structure provides us with an explicit labelling of the PEs (by elements from the group) *and* their incident links (by elements from the generating set); if we can encode the group elements efficiently then this provides us with a natural addressing scheme for the network.

Secondly, several advantages ensue from the fact that Cayley graphs are *symmetric* (*i.e.*, they are vertex-transitive). Symmetry can simplify algorithm design since the network "looks the same" from each PE. Thus, for example, once we have designed a routing algorithm for a single PE, then we have effectively designed a routing algorithm for the entire network. We also conjecture that low-congestion permutation routing is facilitated by the presence of symmetry (see Chapter 3), since there are no obvious "hot-spots" in the network. Finally, symmetry can simplify the discovery of substructures of a network. For example, if there exists a binary tree rooted at some vertex then, by symmetry, there is an isomorphic binary tree rooted at every other vertex in the network (the trees will not necessarily be disjoint).

6. Mapping Problems. Group-theoretic structure may be utilized to find efficient emulations of logical networks on physical networks. In particular, the discovery of uniform mappings (respecting group structure) can simplify the process of transforming an algorithm from a guest network to a host network, or mapping the logical structure of a parallel algorithm (specified by a directed acyclic graph) to a physical network. (This latter idea is completely unexplored territory.)

1.3.2 Previous Work on Cayley Networks

The algebraic study of parallel networks and computation is a very young field. The first authors to take a group-theoretic approach to the construction of parallel network topologies were Akers and Krishnamurthy [3,4]. They focused on Cayley graphs specified by a set of generating permutations, and in particular, they suggested the classes of star

graphs and pancake graphs as candidate topologies. They analyzed several computationally important graph-theoretic properties of these families such as degree, diameter, connectivity and the presence of large complete binary trees (for performing parallel prefix computations). By recognizing that point-to-point routing on any Cayley network of the symmetric group reduced to a sequential sorting problem, they provided efficient routing algorithms for several classes of Cayley networks.

Annexstein, Baumslag and Rosenberg [10] extended the framework of Akers and Krishnamurthy through the introduction of a class of graphs they called *group-action graphs*[2]. The authors found useful relationships, both structural and computational, between group-action graphs and certain "associated" Cayley graphs. The work presented in this thesis builds on the results and the framework laid out in [10].

Faber [44] studied the problem of broadcasting in Cayley networks, deriving a general condition on a group that insured the existence of an optimal broadcast algorithm. Most recently, Blaha [28] improved Faber's method for broadcasting on a Cayley network and studied several routing and reliability problems for Cayley networks equipped with *strong generating sets.*

Fellows [45], in his doctoral thesis, exploited the algebraic structure of graphs to investigate the problem of encoding one graph in another. Encodings of graphs can be thought of as one-to-many embeddings. He mainly concentrated on hypercubes, complete graphs, line graphs and meshes.

Finally, we mention that the use of algebraic properties of graphs has been pursued by Rosenberg in the study of addressing schemes for data structures (see, for example, [90] or [91]). This work had similar motivations to our own, namely, the exploitation of algebraic structure to infer computational capabilites. One important difference, however,

[2]These graphs are known in the group-theoretic literature as *Schreier coset graphs*, and, in this thesis, we refer to them as *right coset graphs.*

is that Rosenberg's work was based on monoids rather than groups, thus leading to quite disparate results.

1.3.3 Network Constructions Based on Cayley Graphs.

The aforementioned research represents the only work (to our best knowledge) in which computational properties of Cayley networks have been studied in a general setting. However, a second area which has received considerable attention is that of using Cayley graphs to construct networks with specific graph-theoretic properties. In particular, Cayley graphs have been used to construct various families of *expander graphs* and *extremal graphs*. We outline some of the main results below.

Expander Graphs. Several authors [6,74,76] have constructed families of *expander* graphs using Cayley graphs of simple groups. Expander graphs are important for optimal parallel sorting algrithms. These results were complemented by Annexstein and Baumslag [9] who showed that any "uniform" family of Cayley graphs of solvable groups could never yield expander graphs.

Extremal Graphs. The aim of this work is to discover graphs with very low diameter with respect to the number of vertices in the graph. This problem is important for VLSI circuit design and for the construction of efficient parallel networks. Firstly, Carlsson et al [34] suggested a generalization of the cube-connected cycles network based on Cayley graphs of the wreath product of cyclic groups equipped with "random" generating sets. Their new networks improved on the diameter of the cube-connected cycles in certain cases. Campbell et al [33] found by exhaustive search that certain Cayley graphs of linear groups are very good extremal graphs; for a fixed degree and diameter they found graphs with a large number of vertices. In several cases, these results supply the densest graphs presently known (the optimal bound for the maximum number of vertices that a graph of a given diameter and degree can contain is known as the Moore bound, which is unattainable except in a finite number of special cases – see, for example, [14]).

It is natural to consider the diameter problem for Cayley graphs with respect to underlying group structure. Babai et al [12] have shown that every non-abelian simple group has a generating set of constant size for which the resulting Cayley graph is of logarithmic diameter and the authors conjecture that *every* non-abelian simple group has a generating set for which the resulting Cayley graph is an expander. Annexstein and Baumslag [9] discovered asymptotic bounds for the diameter of Cayley graphs of abelian and nilpotent groups, proving that they are always algebraic in their degree. See the paper by Babai et al [11] for a comprehensive survey of results in this area.

A fair amount of attention has been devoted to studying the diameter of Cayley graphs of cyclic groups [24,63,61,62,102]. Hsu and Jia [63] have precisely determined the densest possible graphs (in this class) in certain cases and have derived approximate upper bounds in the general case. (For a directed Cayley graph of a cyclic group with 2 generators, and a given value for its diameter they derive an expression for the maximum number of vertices that the graph can contain.) For more information detailing the results in this area, we refer the reader to the survey article by Bermond, Comellas and Hsu [23].

1.3.4 Goals of this Research

In summary, the broad goals of this research are threefold. The first is to develop a unified framework through which a wide range of problems can be attacked. Clearly, one of the primary advantages of this is to encourage *generalized* results incorporating large classes of networks. Our second aim is to use the group-theoretic structure of networks to study their individual properties and provide an extra resource for algorithm design. Finally, an important long-term goal of this research is to understand the impact of symmetry (imparted by group-theoretic structure) on parallel computation. In general, we believe that symmetries exhibited by a parallel algorithm can potentially be exploited to simplify the theoretical analysis and physical implementation of the algorithm.

Subject	Chapter(s)
Routing	3,4,7
Fault-Tolerance	4,7
Graph Embeddings	5,6,7
Cayley Graphs and Networks	2,3,4,5,8
Hamiltonian Circuits	8

Table 1.1: Breakdown of chapters in terms of subject headings.

1.4 Thesis Summary

The remainder of the thesis is organized as follows. Chapter 2 presents the formal framework and includes some important results on which the remainder of the thesis is based. The subsequent three chapters concentrate primarily on the study of Cayley networks. Chapter 3 studies the routing problem in two forms. In Section 3.1, we examine the point-to-point routing problem for Cayley networks while Section 3.2 studies off-line permutation routing. Chapter 4 is devoted to the problem of reliability. In Section 4.1, we extend the routing results of the previous section to a scenario in which some of the processing elements of the network are faulty, and in Section 4.2, we present our results on the connectivity of Cayley graphs.

Chapters 5 and 6 are devoted to two embedding problems. The theme of Chapter 5 is on the problem of embedding Cayley graphs into direct product graphs and discusses an application of this problem to the efficient emulation of large virtual Cayley networks on smaller physical networks. In Chapter 6 we present an embedding of the deBruijn network into the hypercube, and discuss a useful generalization of our technique that may be applied to a variety of embedding problems.

In Chapter 7, we present a deterministic algorithm for reconfiguring faulty hypercube networks. Chapter 8 contains our results on the problem of finding Hamiltonian circuits in Cayley graphs. Finally, in Chapter 9, we present our conclusions and discuss some

problems for future work. For readers interested in a breakdown of the thesis in terms of familiar subject headings, we refer them to Table 1.1.

CHAPTER 2

THE FORMAL FRAMEWORK

In this chapter, we present the formal framework for the thesis and introduce some preliminary results. Section 1 covers basic graph-theoretic terminology and describes the model of computation under consideration. Section 2 introduces basic group-theoretic notions and defines several algebraic families of graphs that relate to Cayley graphs. At the end of the chapter, we illustrate some of the ideas in the framework by defining a new class of trivalent networks called *wreathed networks.*

Conventions. Throughout, we adopt the following notation. Graphs are denoted by script letters or by capital greek letters (for algebraically defined graphs) and groups are denoted by capital roman letters. We use an arrow to distinguish a directed graph (also called a *digraph*) from an undirected graph. New terms appear in bold face when they are first defined. For discussion of any unfamiliar terms and further elucidation of the definitions in this chapter, we refer the reader to the books by Hall [55] and Bollobas [31].

2.1 Graphs and Networks

Throughout the thesis, we focus on synchronous *Multiple Instruction Multiple Data* (MIMD) message-passing machines, although some of our results also apply to *Single Instruction Multiple Data* (SIMD) machines. Formally, a parallel network of this type is modelled by a graph, $\mathcal{N} = (V, E)$, where:

- The vertex set V represents a set of autonomous processing elements (PEs). Each PE consists of a processor and a local memory.

- Each edge is a bidirectional communication link connecting a pair of processing elements.

We assume that the network operates in a *synchronous* manner in which local computation phases alternate with communication phases. During a communication phase, each PE sends and receives a unit of data from its neighbors. Each unit of data is called a *message*, or sometimes a *packet*. Two possible models of communication arise (see [94]). In the **multi-port** model, each processor can use all of its ports in a given communication step whereas in the **single-port** model only one of a processor's ports may be active during a communication step.

2.1.1 Graph-Theoretic Terminology

Almost without exception, the graphs in this thesis are finite, undirected and simple (*i.e.*, without self-loops). Let $\mathcal{G}$ be a connected graph. The **degree** of a vertex is the number of edges incident to it. If all the vertices of a graph have the same degree then the graph is said to be **regular**. In this case we denote its common degree by $\rho(\mathcal{G})$.

The **distance** between two vertices of $\mathcal{G}$ is the length of a shortest path[1] connecting them. The **diameter** of $\mathcal{G}$, denoted by $\mathrm{diam}(\mathcal{G})$, is the maximum distance between a pair of vertices, taken over all possible pairs. If H is any subset of the vertices of $\mathcal{G}$, then the maximum distance in $\mathcal{G}$ between any two vertices of H is denoted by $\mathrm{diam}_{\mathcal{G}}(H)$.

Define the **direct product** of two graphs, $\mathcal{G} = (V_1, E_1)$ and $\mathcal{H} = (V_2, E_2)$, denoted by $\mathcal{G} \times \mathcal{H}$, as follows. The vertex set is the cartesian product $V_1 \times V_2$. There is an edge between (v_1, v_2) and (v_1', v_2') when $v_1 = v_1'$ and $(v_2, v_2') \in E_2$, or $v_2 = v_2'$ and $(v_1, v_1') \in E_1$. $\mathcal{G}$ and $\mathcal{H}$ are called the **factors** of $\mathcal{G} \times \mathcal{H}$. Notice that $\mathcal{G} \times \mathcal{H}$ consists of $|V_2|$ copies of $\mathcal{G}$ connected by $|V_1|$ copies of $\mathcal{H}$, arranged in a grid-like fashion. By convention, each copy of $\mathcal{H}$ is called a **row** and each copy of $\mathcal{G}$ is called a **column**.

[1] A path in a graph is a sequence of abutting edges.

2.1.2 Important Parallel Networks

We now define several familiar "benchmark" topologies to which we repeatedly refer as examples and case studies throughout the thesis. (See Figure 2.1.)

Ring and Line graphs. Let $\mathcal{R}_n$ denote an n-cycle and let $\mathcal{L}_n$ denote an n-vertex line. (By convention, $\mathcal{R}_2$ is a single edge.)

Mesh graphs. The order-n version of this network is denoted by $\mathcal{M}_n$. It is defined as the direct product of two copies of $\mathcal{L}_n$.

Hypercube graphs. The (binary) order-n hypercube (also refered to as an n-dimensional hypercube), denoted by $\mathcal{H}_n$, is the direct product of n copies of $\mathcal{R}_2$. In general, the d-ary n-cube is the direct product of n copies of $\mathcal{R}_d$. An n-ary 2-cube is often called a *toroidal mesh.*

The following four graphs are commonly known as the *butterfly-derivative* graphs.

deBruijn graphs. The (binary) order-n deBruijn graph, denoted by $\mathcal{D}_n$, has a vertex set consisting of the set of all binary n-tuples denoted by $(\beta_0, \beta_1, \ldots, \beta_{n-1})$, where each $\beta_i \in \{0, 1\}$. If β denotes a single bit, let $\overline{\beta}$ denote the complement of β. The edges are of 2 types:

- **shuffle edges:** $(\beta_0, \beta_1, \ldots, \beta_{n-1})$ is adjacent to $(\beta_{n-1}, \beta_0, \beta_1, \ldots, \beta_{n-2})$ and $(\beta_1, \beta_2, \ldots, \beta_{n-1}, \beta_0)$.
- **shuffle-exchange edges:** $(\beta_0, \beta_1, \ldots, \beta_{n-1})$ is adjacent to $(\beta_1, \ldots, \beta_{n-1}, \overline{\beta_0})$ and $(\overline{\beta_{n-1}}, \beta_0, \ldots, \beta_{n-2})$.

We mention that the definition of a deBruijn graph may be generalized by using the set of all d-ary strings as the vertex set, for any $d \geq 2$. In this case, each vertex $(d_0, d_1, \ldots, d_{n-1})$ is adjacent to $(d_1, d_2, \ldots, x)$ for each $x \in \{0, \ldots, d-1\}$. These graphs are commonly called the *generalized deBruijn graphs.*

Butterfly graphs. Define the order-n butterfly network, denoted by $\mathcal{B}_n$, as follows. The vertex set consists of the set of pairs

$$\cdot\langle \ell; B \rangle,$$

where ℓ is an integer between 0 and $n-1$ and B is an n-bit integer. (Thus, there are $n2^n$ vertices in $\mathcal{B}_n$.) The set of all vertices with a fixed first component is called a **level** of the butterfly. The set of all vertices with a fixed second component is called a **column**. There is an edge between $\langle \ell_1; B_1 \rangle$ and $\langle \ell_2; B_2 \rangle$ when either

- $B_1 = B_2$ and $|\ell_1 - \ell_2| \bmod n = 1$ or
- $|\ell_1 - \ell_2| \bmod n = 1$ and B_2 differs from B_1 in the ℓ_1-th bit.

There are two common networks based on the butterfly topology. If the last level of the butterfly does not wrap around back to the first, then the resultant network is called the FFT network[2]. Two copies of an FFT network connected back to back (the last level of one copy is joined to the last level of a second copy) forms the *Beneš* network.

Shuffle-exchange graphs. The order-n shuffle-exchange graph, denoted by $\mathcal{S}_n$, has a vertex set consisting of the set of all binary n-tuples. As in the deBruijn graph the edges are of two types. The first type are the shuffle edges. The second are the **exchange** edges in which $(\beta_0, \beta_1, \ldots, \beta_{n-1})$ is adjacent to $(\beta_0, \beta_1, \ldots, \overline{\beta_{n-1}})$, where $(\beta_0, \beta_1, \ldots, \beta_{n-1})$ represents an arbitrary binary n-tuple.

Cube-connected cycles graphs. We denote the order-n cube-connected cycles graph by $\mathcal{C}_n$. The vertex set is identical to that of the butterfly graph. There is an edge between $\langle \ell_1; B_1 \rangle$ and $\langle \ell_2; B_2 \rangle$ when either

- $B_1 = B_2$ and $|\ell_1 - \ell_2| \bmod n = 1$, or
- $\ell_1 = \ell_2$ and B_2 differs from B_1 in the ℓ_1-th bit.

[2]The name is derived from the fact that the adjacencies of this graph reflect the movement of data during a *fast fourier transform* computation.

2.1.3 Graph Embeddings

In order for one network (the host) to simulate the computation of another (the guest), we must provide a means for effecting the logical communication patterns of the guest on the physical host. To this end, we adopt the standard notion of a *graph embedding.* In the study of graph embeddings, it is tacitly assumed that the computational capabilities of the PEs in the host network are commensurate in power with the PEs in the guest network.

Formally, an **embedding** of a graph $\mathcal{G}$ in a graph $\mathcal{H}$ is a pair of maps (ϕ, ρ) where ϕ is an **assignment** of the vertices of $\mathcal{G}$ to the vertices of $\mathcal{H}$ and ρ is a **routing** of the edges of $\mathcal{G}$ (denoted by $E(\mathcal{G})$) to paths of $\mathcal{H}$ such that if (u, v) is an edge of $\mathcal{G}$ then the path $\rho(u, v)$ originates at $\phi(u)$ and terminates at $\phi(v)$ in $\mathcal{H}$. We measure the efficiency of an embedding via the following parameters:

- The **expansion factor** is the ratio

$$\mathsf{exp}(\phi, \rho) = \frac{|\mathcal{H}|}{|\mathcal{G}|}.$$

- The **load factor** is the maximum number of vertices of $\mathcal{G}$ assigned to a single vertex of $\mathcal{H}$:

$$\mathsf{load}(\phi, \rho) = \max_{v \in \mathcal{H}} |\phi^{-1}(v)|$$

- The **dilation** of the embedding is the maximum amount that the routing ρ "stretches" any edge of $\mathcal{H}$:

$$\mathsf{dil}(\phi, \rho) = \max_{(u,v) \in E(\mathcal{G})} \mathsf{length}(\rho(u, v))$$

- The **congestion** of the embedding is the maximum number of edges of $\mathcal{G}$ that ρ routes across a single edge of $\mathcal{H}$:

$$\mathsf{cong}(\phi, \rho) = \max_{e \in E(\mathcal{H})} |\{e' \in E(\mathcal{G}) \ : \ e \in \rho(e')\}|$$

- The **dynamic congestion** of the embedding is the maximum number of messages that must be transmitted over any edge of $\mathcal{H}$ *simultaneously*, when $\mathcal{H}$ simulates a single communication step of $\mathcal{G}$.

In practice, dynamic congestion is overcome by providing a *queuing* mechanism at the PEs of the network.

2.2 Groups and Graphs

Let G be a finite group. A **generating set** for G is a subset S of G (not containing the identity element) with the property that every element of G can be expressed as a product of elements in S (not necessarily uniquely). If S generates G, we write $G = \text{gp}(S)$. We denote the identity element of a group G by e_G (we usually elide the subscript if the context is clear). If the group is additive we elect to use a "0" to represent the identity. The order of an element $s \in G$ is the smallest power k such that $s^k = e_G$ and it is denoted by $o(s)$. If H is a subgroup of G then we write $H \leq G$.

Define the undirected **Cayley graph of G with respect to S**, denoted by $\Gamma(G, S)$, as follows.

- The vertices are the elements of G.
- There is an edge between g and h if $gs = h$ or $hs = g$, for some $s \in S$.

$\Gamma(G, S)$ is easily seen to be regular with $\rho(\Gamma(G, S)) = |S \cup S^{-1}|$ where $S^{-1} = \{s^{-1} : s \in S\}$. Sometimes we regard $\Gamma(G, S)$ as a *directed* graph, denoted by $\vec{\Gamma}(G, S)$. In this case, there is an *arc* from g to h if $gs = h$, for some $s \in S$.

If $H \leq G$ then each set of the form $Hg = \{hg : h \in H\}$ is called a **right coset** of H in G. The set of all right cosets forms a partition of G. Similarly, a set of the form $gH = \{gh : h \in H\}$ is called a **left coset** of H in G. A (left) right **transversal**, T, of H in G is a complete set of representatives for the set of (left) right cosets of H in G, *i.e.*,

T contains one element from each (left) right coset. We denote the representative of an element $g \in G$ (with respect to a fixed transversal) by $\overline{g}$. The **index** of H in G is the number of distinct cosets, and is denoted by $|G : H|$.

G/H denotes the set of all (right) left cosets and is called the **(right) left quotient space.** When the left quotient space coincides with the right quotient space then the subgroup H is said to be **normal** in G and we write $H \lhd G$.

An **automorphism** of a graph is a bijection on the vertex set of the graph that preserves the graph's adjacency. The set of all automorphisms of a graph forms a finite group, called the **automorphism group.** A graph is said to be **vertex-transitive** if the automorphism group of the graph acts transitively on the vertices, *i.e.*, for any pair of vertices x, y, there is an automorphism mapping x to y. We now record some familiar basic properties of Cayley graphs.

Proposition 1 *Every Cayley graph $\Gamma(G, S)$ has the following properties:*

- *$\Gamma(G, S)$ is connected.*
- *$\Gamma(G, S)$ is vertex-transitive.*
- *If H is a non-trivial proper subgroup of G then the induced subgraphs on all of the left cosets of H in G are isomorphic.*

Proof. $\Gamma(G, S)$ is connected since S generates G. To see that $\Gamma(G, S)$ is vertex-transitive consider, for each $g \in G$, the map of the form $\alpha_g : G \to G$ defined by

$$\alpha_g(g') = gg',$$

for $g' \in G$. It is a simple exercise to verify that α_g is an automorphism of $\Gamma(G, S)$. Also, if g_1 and g_2 are arbitrary elements of G then $\alpha_{g_2 g_1^{-1}}$ is an automorphism mapping g_1 to g_2 proving that $\Gamma(G, S)$ is vertex-transitive. The last assertion is proved by noting that each

left coset of H in G can be obtained by multiplying H on the left by any representative of the coset. Since multiplication on the left defines an automorphism of $\Gamma(G, S)$, the induced subgraphs on all left cosets must be isomorphic. □

Two subclasses of Cayley graphs are of particular interest, mainly because they subsume several benchmark topologies. Loosely speaking, these graphs are ones for which there is no redundancy in the generating set. More precisely, S is said to be a **minimal** generating set for G if S generates G but no proper subset of S also generates G. More generally, we define S to be a **quasi-minimal** generating set for G if there exists an ordering of S, say $s_1, s_2, \cdots, s_k$, such that $s_i \notin \text{gp}(s_1, s_2, \cdots, s_{i-1})$ for $2 \leq i \leq k$. The following example shows that not every quasi-minimal generating set for a group G is minimal. Consider the set of permutations

$$S = \{(1,2), (1,2,3), \cdots, (1,2,3,\cdots,n)\}$$

for any $n \geq 2$, which generates the symmetric group on n symbols, denoted by $\text{Sym}(n)$. This is clearly a quasi-minimal generating set for $\text{Sym}(n)$ for all n but contains a subset of size 2, namely $\{(1,2), (1,2,\cdots,n)\}$, which also generates $\text{Sym}(n)$ (see [40]).

Notice that, by Proposition 1, both minimal and quasi-minimal Cayley graphs are *scalable* in the sense that they recursively consist of copies of smaller Cayley graphs of the same variety (see Section 2.2.2 for a description of how the copies join together).

2.2.1 Coset Graphs

Given a Cayley graph and a subgroup of the underlying group of the Cayley graph, we can construct two natural "quotient" graphs.

Right coset graphs. If H is a subgroup of G then we define the **right-coset graph** of G with respect to H, denoted by $\Gamma_R(G/H, S)$, as follows:

- The vertex set is composed of the *right* cosets of H in G.

- There is an edge between Hg_1 and Hg_2 if $Hg_1s = Hg_2$ or $Hg_2s = Hg_1$ for some $s \in S$.

Note that this definition yields the definition of a Cayley graph when H is the trivial subgroup.

Left coset graphs. The following construction due to Sabidussi [95] (also, see [112]) performs a central role in our study of Cayley graphs. Let $G = \text{gp}(S)$ and suppose that H is a non-trivial subgroup of G. The **left coset graph**[3], denoted by $\Gamma_L(G/H, S)$, has a vertex set consisting of the *left* cosets of H in G and an edge set comprising all pairs of distinct cosets, (g_1H, g_2H), for which there exists an $s \in S$ such that

$$g_1h_1s = g_2h_2$$

for some $h_1, h_2 \in H$.

Proposition 2 (Theorem 2, [95]) *The left coset graph, $\Gamma_L(G/H, S)$, is vertex-transitive for any $H \leq G$.*

Proof. Since multiplication on the left induces an automorphism of $\Gamma(G, S)$ and two distinct left cosets are disjoint, the result follows. □

We remark that the class of left-coset graphs characterizes vertex-transitive graphs [95], a result known as Sabidussi's representation theorem.

2.2.2 Structure Theorems for Coset Graphs

In this section, we present two fundamental results which we require throughout the thesis. These results are the basic tools of the formal framework, and in Section 2.3, we illustrate their use in a specific example.

[3]In [112], this graph is called a *group-coset graph*.

Right Coset Graphs

Let $G = \text{gp}(S)$ and suppose $H \leq G$. The following theorem exposes an important graph-theoretic relationship between $\Gamma_R(G/H, S)$ and $\Gamma(G, S)$.

Theorem 1 ([10]) *Let $\mathcal{T}$ be any subgraph of $\Gamma_R(G/H, S)$ which is a tree. Then there are $|H|$ disjoint copies of $\mathcal{T}$ in $\Gamma(G, S)$.*

Proof. Suppose T is a right transversal for H in G. Consider multiplying T on the left by some $h \in H$. It is easily seen that we obtain another right transversal for H in G which is *disjoint* from T. Hence, since multiplication on the left is an automorphism of $\Gamma(G, S)$, there are $|H|$ disjoint copies of $\mathcal{T}$ in $\Gamma(G, S)$. □

In particular, Theorem 1 implies that we can construct a spanning tree of $\Gamma(G, S)$ by taking the union of any spanning tree of the Cayley graph of a subgroup $H = \text{gp}(S')$, where $S' \subset S$, together with $|H|$ copies of a spanning tree of $\Gamma_R(G/H, S)$. In Figure 2.2, we give an example of this theorem by drawing the order-3 *star graph*[4]. This graph is a Cayley graph of Sym(4), the symmetric group on four points, $\{1, 2, 3, 4\}$, with a generating set consisting of the three transpositions that interchange 1 with x, for $x = 2, 3,$ and 4. (Note that we write permutations in point-wise notation in the figure.) The subgroup H that we use is Sym(3), generated by the two transpositions that interchange 1 with each of 2 and 3.

Left Coset Graphs

Let $H = \text{gp}(S_H)$, $S = S_H \cup \{s\}$ and suppose $s \notin H$. Note that if S_H is a quasi-minimal generating set for H then S is a quasi-minimal generating set for $G = \text{gp}(S)$. We now investigate the graph-theoretic structure of $\Gamma(G, S)$ in relation to the left coset graph,

[4]We refer the reader to Chapter 3 for the general definition of this class of graphs.

$\Gamma_L(G/H, S)$. Firstly, observe that all of the edges between left cosets arise from s-edges since, by definition of S, edges arising from the other generators lie within $\Gamma(H, S_H)$. The following theorem sheds light on the structure of quasi-minimal Cayley graphs.

Theorem 2 *The number of edges in $\Gamma(G, S)$ that collapse to form each edge of $\Gamma_L(G/H, S)$ is a fixed constant.*

Proof. Without loss of generality, we may restrict ourselves to the edges emanating from H since Cayley graphs are vertex-transitive. Let

$$K = \{h \in H : hs = sh' \text{ for some } h' \in H\}.$$

Then it is evident that $K = sHs^{-1} \cap H$ and thus K is a subgroup of H. Notice that the edge (H, sH) of $\Gamma_L(G/H, S)$ is formed by collapsing at least $|K|$ edges of $\Gamma(G)$.

We now show that the set of s-edges emanating from each coset of K in H collapses to form a *distinct* edge of $\Gamma_L(G/H, S)$. Choose two distinct cosets of K in H, say $r_1K \neq r_2K$ (for some $r_1, r_2 \in H$) and consider the cosets (of H in G) in which these sets of group elements end up after multiplication by s:

$$\begin{aligned} r_1Ks &= r_1sH' \\ r_2Ks &= r_2sH' \end{aligned}$$

for some $H' \subseteq H$.

CLAIM: r_1s and r_2s are in distinct cosets of H in G.

Suppose the claim is false. Then there exists an $h \in H$ such that

$$(r_2s)^{-1}r_1s = h,$$

and hence

$$r_2^{-1}r_1s = sh.$$

This implies that $r_2^{-1}r_1 \in K$, *i.e.*, r_1 and r_2 are in the same coset modulo K, contradicting the choice of r_1 and r_2. □

This theorem shows that Cayley graphs are formed in a very regular fashion from left coset graphs and induced Cayley graphs of subgroups. Let d_e denote the index of K in H and let d_i denote the size of K. If s is of order 2, then

$$\rho(\Gamma_L(G/H, S)) = d_e,$$

and the number of edges of $\Gamma(G, S)$ collapsing to form a single edge of $\Gamma_L(G/H, S)$ equals d_i. In Chapter 4, we will closely examine the structure of $\Gamma(G, S)$ when s is not of order 2. However, in the sequel, we adopt the following terminology:

- d_i is called the **internal degree** of $\Gamma_L(G/H, S)$.
- d_e is called the **external degree** of $\Gamma_L(G/H, S)$.
- The set of *all* edges of $\Gamma(G, S)$ connecting a pair of left cosets (of H in G) is called a **bundle**. Note that the size of a bundle is always at least $|K|$.

Finally, we observe that if $H \lhd G$ then the left and right coset graphs are isomorphic. In Figure 2.3, we provide a schematic depiction of the structure theorem for left coset graphs.

2.2.3 Semidirect Products and Wreath Products

Denote the group of integers modulo k by Z_k and the direct product of n copies of a group G by G^n. Suppose there are two subgroups H and N of a group G where N is normal in G and the following two conditions hold:

1. $G = HN$
2. $H \cap N = \{e\}$.

Then G is said to be the (internal) **semidirect product of N by H**, and we denote it by $H \otimes_{sd} N$. In this case, the quotient group G/N is isomorphic to H, which we write as $G/N \simeq H$. Every element of a semidirect product can be uniquely expressed as a pair (h, n) where $h \in H$ and $n \in N$.

Suppose S_1 is a generating set for H and S_2 is a generating set for N. We form the *standard* generating set for the semidirect product of N by H, denoted by $\mathbf{S}$, as follows.

$$\mathbf{S} = \{(s_1, e) : s_1 \in S_1\} \cup \{(e, s_2) : s_2 \in S_2\}.$$

The wreath product[5] of G by Z_n, denoted by $G \wr Z_n$, is defined to be a semidirect product of G^n by Z_n. Every element π of $G \wr Z_n$ is represented by a pair

$$\pi = \langle \alpha; \beta_0, \beta_1, \cdots, \beta_{n-1} \rangle,$$

where $\alpha \in Z_n$ and $(\beta_0, \beta_1, \cdots, \beta_{n-1}) \in G^n$. α is termed the first component of $G \wr Z_n$ and $(\beta_0, \beta_1, \cdots, \beta_{n-1})$ is termed the second component. Multiplication of π with another element $\pi' = \langle \alpha'; \beta'_0, \beta'_1, \cdots, \beta'_{n-1} \rangle$ occurs as follows:

$$\pi \cdot \pi' = \langle \alpha + \alpha'; \beta_{\alpha'} \cdot \beta'_0, \beta_{\alpha'+1} \cdot \beta'_1, \ldots, \beta_{\alpha'-1} \cdot \beta'_{n-1} \rangle,$$

where all subscripts are evaluated modulo n and "$\cdot$" represents multiplication in G. In more descriptive terms, the first component of the product is the sum of the first components of π and π' while the second component is formed by performing α' left circular cyclic shifts of the second component of π, followed by a component-wise multiplication of the second component of π' with the "shifted" second component of π.

Generating sets for wreath products. Let S generate G. The **standard generating set** for $G \wr Z_n$ is defined as

$$\mathbf{S} = \{\langle 1; e, e, \ldots, e \rangle\} \cup \{\langle 0; e, e, \ldots, e, s \rangle : s \in S\}.$$

[5]This is commonly called the *standard wreath product* (see [89]).

We now introduce a second generating set for wreath products called the **counting generating set**[6], which is defined as

$$\mathbf{C} = \{\langle 1; e, e, \ldots, e\rangle\} \cup \{\langle 1; e, e, \ldots, s\rangle : s \in S\}.$$

Let Z_2^n be equipped with the standard generating set, which consists of the set of all bit vectors of length n that have, for each i $(0 \leq i \leq n-1)$, a 1 in the i-th place and a 0 elsewhere. We can algebraically express the topologies of most of the parallel networks described in Section 2.1.2 according to the following proposition.

Proposition 3 *The following graph isomorphisms hold:*

- $\mathcal{R}_n = \Gamma(Z_n, \mathbf{S})$
- $\mathcal{B}_n = \Gamma(Z_2 \wr Z_n, \mathbf{C})$
- $\mathcal{D}_n = \Gamma_R(Z_2 \wr Z_n/Z_n, \mathbf{C})$
- $\mathcal{C}_n = \Gamma(Z_2 \wr Z_n, \mathbf{S})$
- $\mathcal{S}_n = \Gamma_R(Z_2 \wr Z_n/Z_n, \mathbf{S})$
- $\mathcal{H}_n = \Gamma(Z_2^n, \mathbf{S})$

Proof. The first isomorphism is obvious. The next four isomorphisms were established in [10]. For the group-theoretic description of the hypercube, notice that the graph-theoretic direct product of two Cayley graphs $\Gamma(G_1, S_1)$ and $\Gamma(G_2, S_2)$ is isomorphic to the Cayley graph $\Gamma(G_1 \times G_2, \mathbf{S})$. (This fact has been noted in several places in the literature – see, for example, [110] or [4].) Since the hypercube is a direct product of 2-cycles (*i.e.*, single edges) and each 2-cycle is (isomorphic to) $\Gamma(Z_2, \mathbf{S})$, the result follows. □

[6]This generating set was discovered by Annexstein et al [10] and independently by Fellows [45] and Carlsson et al [34].

2.3 An Illustration - Networks Based on Wreath Products.

In this section, we define a new class of trivalent networks called *wreathed networks* and use them to illustrate some of the ideas contained in the formal framework. Let $n > 1$ be an integer, let k be a divisor of n, and consider the following 2 elements of $Z_2 \wr Z_n$:

$$\begin{aligned} \tau_n &= \langle 1; 0, 0, \cdots, 0 \rangle \\ \sigma_{n,k} &= \langle 0; s_{n,k} \rangle, \end{aligned}$$

where $s_{n,k}$ is a length-n bit string of the form

$$\overbrace{0, \cdots, 0, 1}^{k}, \overbrace{0, \cdots, 0, 1}^{k}, \cdots .$$

In other words, $s_{n,k}$ is a bit string composed of n/k copies of the length-k bit string $0, \cdots, 0, 1$. We say that $s_{n,k}$ has **period** k, since the string obtained by performing k left circular shifts results in an identical string. Let $G_{n,k} = \mathrm{gp}(\tau_n, \sigma_{n,k})$. The **wreathed network** of order-(n, k), denoted by $\mathcal{W}_{n,k}$, is defined to be the Cayley graph $\Gamma(G_{n,k}, \{\tau_n, \sigma_{n,k}\})$. In Figure 2.4, we depict the order-$(6, 3)$ wreathed network. Clearly, $\mathcal{W}_{n,k}$ has $n2^k$ vertices. The following theorem investigates the size of each bundle, and the structure of the left and right coset graphs for wreathed networks.

Theorem 3 *Define $H_{n,k} = \mathrm{gp}(\tau_n)$. The following three assertions hold for $\mathcal{W}_{n,k}$.*

1. *The size of each bundle is equal to n/k.*
2. *$\Gamma_R(G_{n,k}/H_{n,k}, \{\tau_n, \sigma_{n,k}\}) = \mathcal{S}_k$.*
3. *$\Gamma_L(G_{n,k}/H_{n,k}, \{\tau_n, \sigma_{n,k}\}) = \mathcal{H}_k$.*

Proof (of Assertion 1). Since the order of $\sigma_{n,k}$ is 2, determining the bundle size of $\mathcal{W}_{n,k}$ is equivalent to determining the size of the subgroup K defined in Theorem 2. (In

this case, the size of each bundle *equals* the size of K.) By definition, we have

$$K = \{\xi \in H_{n,k} : \xi\sigma_{n,k} = \sigma_{n,k}\xi',\ where\ \xi' \in H_{n,k}\}.$$

Firstly, it is evident that for any $\xi \in H_{n,k}$,

$$\xi\sigma_{n,k} = \langle i; s_{n,k}\rangle,$$

for some $i \in Z_n$. Secondly, observe that $\sigma_{n,k}\xi'$ is equal to an element of the form $\langle i; s_{n,k}\rangle$, for some $i \in Z_n$, precisely when the first component of ξ' is equal to a multiple of k. This may be further elucidated as follows: If the first component of ξ' is a multiple of k, then the second component of $\sigma_{n,k}$ is rotated k times when we perform the multiplication $\sigma_{n,k}\xi'$. Since the second component of $\sigma_{n,k}$ has period equal to k, the second component of the resultant product will equal $s_{n,k}$ exactly when ξ' is a multiple of k. This occurs n/k times, and hence the size of each bundle is n/k.

Proof (of Assertion 2). Firstly, note that the set of elements in any given right coset has a fixed second component. In fact, the reader may easily verify that each of the 2^k possible values for the second component completely characterizes a single right coset of $H_{n,k}$ in $G_{n,k}$. Let B denote the length k bit string representing a particular right coset (the reader may find it useful to picture each distinct value of B as a distinct vertex in S_k). The reason we need only k bits is a consequence of the fact that the second component of any element in $G_{n,k}$ has period k. (In other words, the second component of a coset element represented by B consists of the concatenation of n/k copies of some bit string B.)

We now consider the action (*i.e.*, the effect of right multiplication) of the generators τ_n and $\sigma_{n,k}$ on the right cosets. Consider the action of τ_n on the right coset represented by B. From the definition of multiplication in the wreath product, we see that the effect is to "move us" to another right coset represented by the *left circular shift* of B. Taken over all cosets, this produces the shuffle edges of S_k.

Finally, consider the action of $\sigma_{n,k}$ on the right coset represented by B. Again, from the definition of the wreath product, the effect is to "move us" to a right coset represented by B with the low-order bit complemented. Taken over all right cosets, this produces the *exchange* edges of S_k. Hence, the assertion is proved.

Proof (of Assertion 3). Consider any fixed left coset of $H_{n,k}$ in $G_{n,k}$. Let us choose as a representative the (unique) element that has a "0" in the first component, and denote it by B. Encode each representative with the length-k bit string that completely describes the representative's second component. (Observe that the second component is comprised of n/k copies of the same length-k bit string.) For convenience, we let $\tilde{B}$ denote this encoding.

We now examine the structure of the edges of $\Gamma_L(G_{n,k}/H_{n,k}, \{\tau_n, \sigma_{n,k}\})$. Since $\tau_n \in H_{n,k}$, multiplication by τ never forms an edge of $\Gamma_L(G_{n,k}/H_{n,k}, \{\tau_n, \sigma_{n,k}\})$. (It creates a self-loop at every vertex.) Thus, we turn our attention to $\sigma_{n,k}$. Consider an arbitrary left coset, whose representative is B. Any other element of the same left coset can be obtained from B by multiplying B (on the right) by some power of τ_n, since τ_n generates $H_{n,k}$.

We now need to determine which left cosets can be reached from the one represented by B. This amounts to examining all of the products of the form $B\tau_n^i\sigma_{n,k}$ (for $0 \leq i \leq n-1$). Because of our choice of representatives for the left cosets, the representative of $B\tau_n^i\sigma_{n,k}$ is the element $B\tau_n^i\sigma_{n,k}\tau_n^{-i}$. Let us examine what happens to the k low-order bits of the second component of B while performing this multiplication. (In other words, we are examining what happens to $\tilde{B}$.) Firstly, τ_n^i performs i *left* circular shifts on it; secondly, $\sigma_{n,k}$ complements the low-order bit, and finally τ_n^{-i} performs i *right* circular shifts. Thus, this product maps the representative B to a left coset whose encoding is the same as that for B except that the i-th bit from the left (modulo k) has been complemented. This implies that $\tilde{B}$ is adjacent to all $\overline{B'}$ differing in a single bit, proving

that $\Gamma_L(G_{n,k}/H_{n,k}, \{\tau_n, \sigma_{n,k}\})$ is isomorphic to the k-dimensional hypercube. □

Remarks. The wreathed networks are a generalization of the cube-connected cycles networks [86] since we have $\mathcal{W}_{n,n} = \mathcal{C}_n$ (see [10]). Assertion 2 of Theorem 3 was originally proved in [10] for this special case. Finally, the reader may observe that $\mathcal{W}_{n,k}$ can be constructed by joining n/k copies of $\mathcal{W}_{k,k}$ that have been "unfolded" (*i.e.*, edges on the last level are severed) – see Figure 2.4.

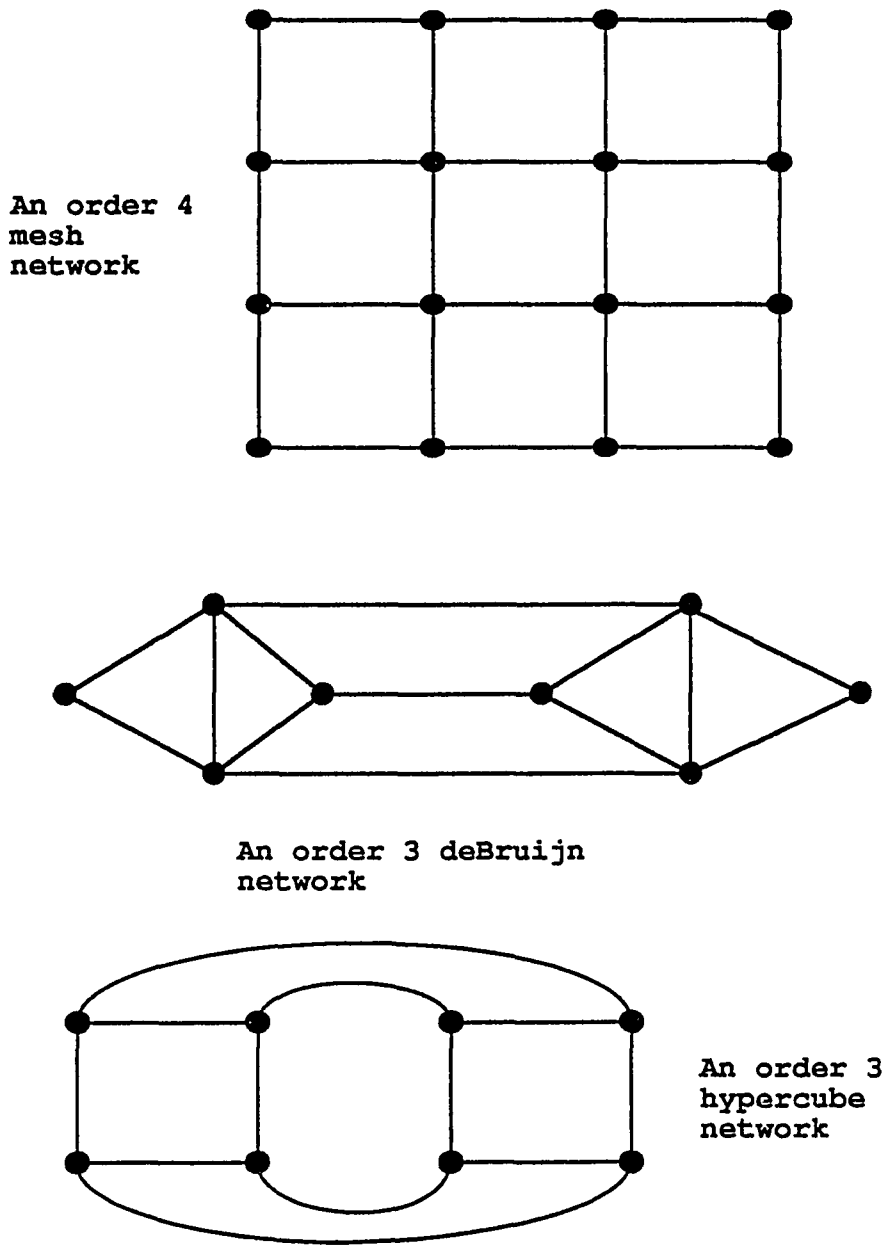


Figure 2.1: Small versions of some common network topologies

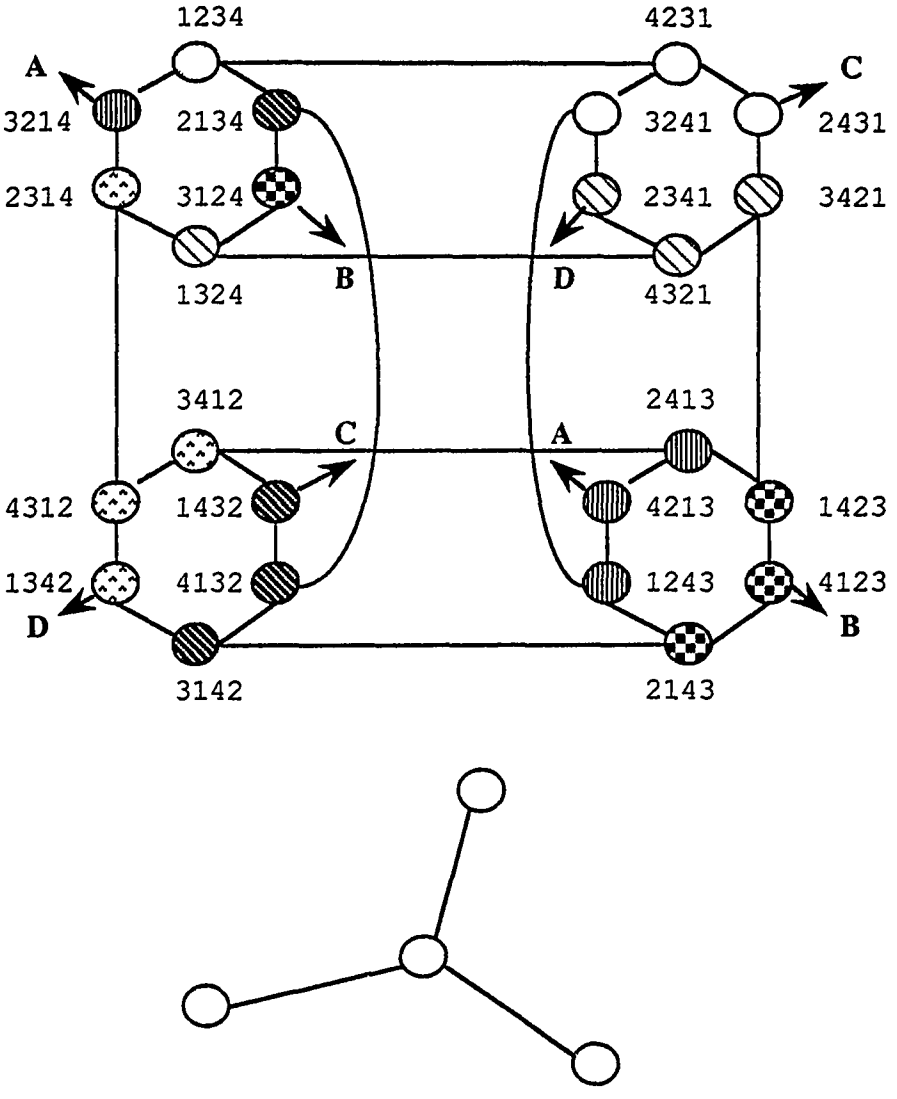


Figure 2.2: The order-3 star graph (top), and a depiction of its decomposition into copies of a spanning tree of a right quotient graph (bottom)

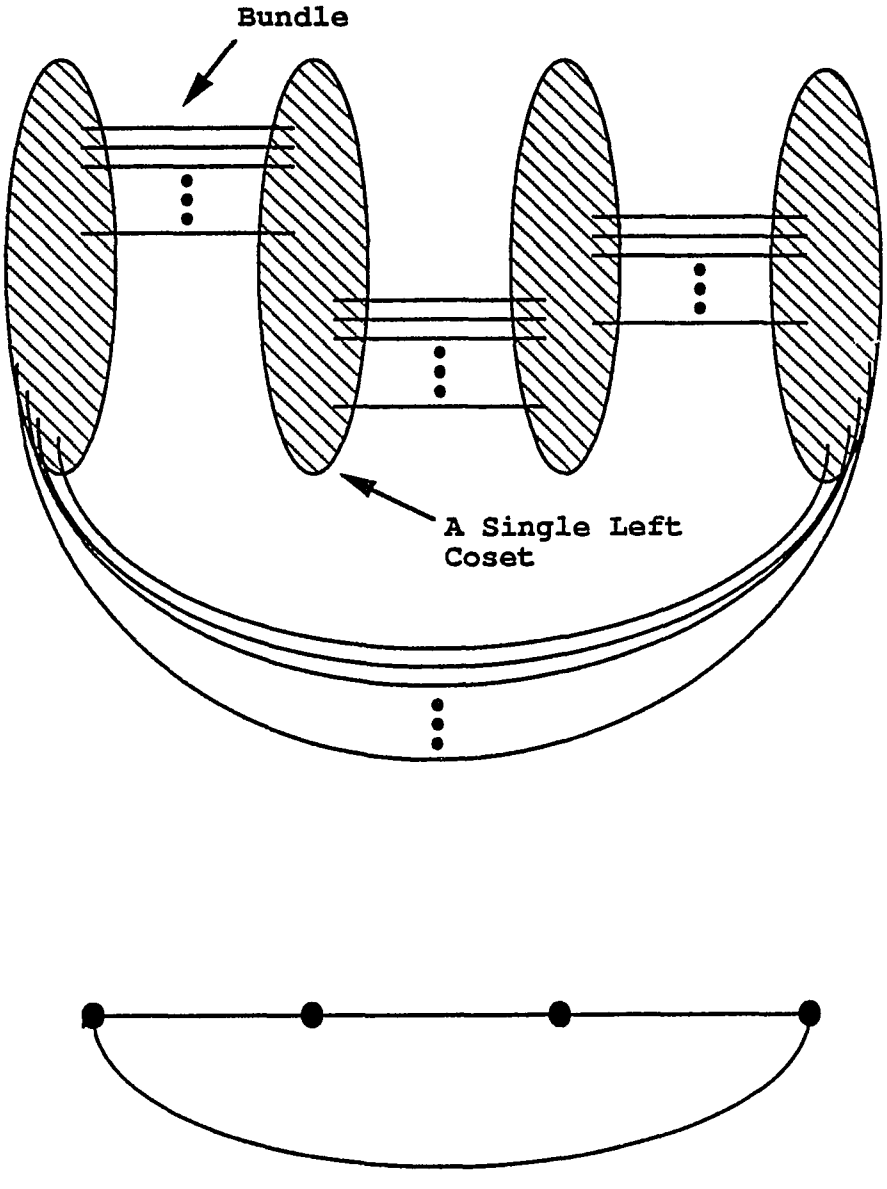


Figure 2.3: Schematic depiction of a quasi-minimal Cayley graph (top) and its left quotient graph (bottom)

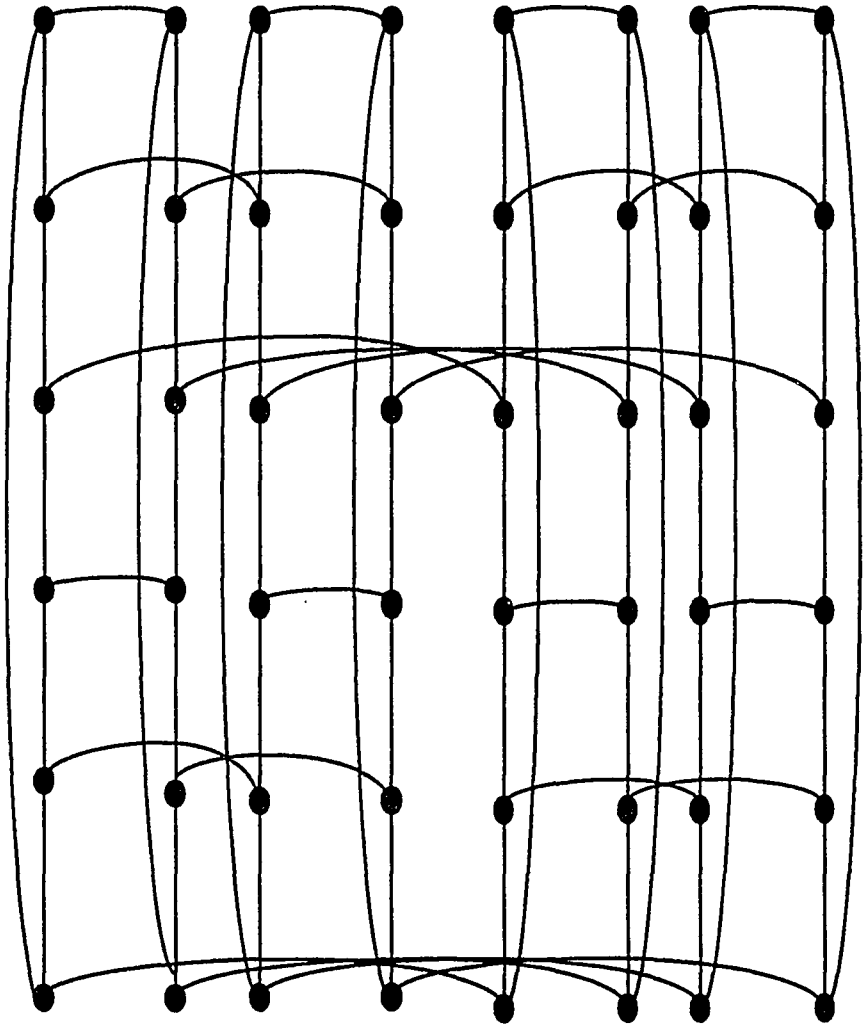


Figure 2.4: The order-$(6, 3)$ wreathed network, $\mathcal{W}_{6,3}$

CHAPTER 3

Routing Problems for Cayley Networks

In this chapter, we address two fundamental problems concerning routing in parallel networks. The first section addresses *point-to-point routing* in Cayley networks. There are two central issues regarding point-to-point routing in a network; namely, how to choose a labelling for the PEs of the network and how to use the labelling to facilitate routing. We discuss how an efficient labelling of a Cayley network arises from the group structure and we demonstrate how this can lead to an efficient routing mechanism.

In the second section, we present a general strategy for finding efficient *permutation routes* in an off-line setting for parallel networks. The strategy is applicable to a wide spectrum of topologies, including all of the benchmark topologies defined in Chapter 2.

3.1 Point-to-Point Routing in Cayley Networks

Our first aim is to show how a Cayley network inherits a labelling from the underlying group structure. The primary purpose of a labelling scheme is to facilitate routing, and in the latter half of this section, we develop two schemes for routing based on our labelling methods. We illustrate the ideas by including a group-theoretic description of how the familiar labelling of the butterfly network arises directly from the group structure.

3.1.1 Symmetric Routing

We can partition the problem of designing an efficient point-to-point routing algorithm for a network into two distinct tasks (see, for example, [96]):

1. Find an appropriate labelling, $\mathbf{L}$, of $\mathcal{N}$, *i.e.*, find an assignment of *labels* (bit strings) to the PEs (vertices) and ports (edges) of $\mathcal{N}$.

2. For each PE a, construct a function

$$\rho_a : \mathbf{L} \rightarrow \mathsf{ports}(a)$$

such that if a message arrives at a and is destined for $\ell \in \mathbf{L}$, then unless $\ell = \mathsf{label}(a)$ (in which case the message has arrived at its destination), a sends the message out of port $\rho_a(\ell)$.

In general, before a message is sent, the originating PE appends some routing information, including, at least, the label of the destination PE. We may distinguish two types of routing schemes:

- **Explicit** routing schemes store the functions ρ_a (for each a) explicitly in a local table.

- **Implicit** routing schemes use the labels themselves to find the next step in a route, *i.e.*, the labels are no longer passive addresses; instead, they encode useful routing information.

Suppose we wish to design an explicit routing scheme for a Cayley network which has G as its underlying group. At each PE, we store a routing table such that, when a message arrives, the PE looks up the appropriate outgoing port through which the message must proceed, using the message's destination as an index. Since there is a path from a to x in a Cayley graph with identical edge labels (*i.e.*, the generators are identical) as a path from 1 to $a^{-1}x$, we may use the *same* routing table at PE a as the one used by the identity PE, except that the entries of the table must be "permuted" by a^{-1}. More precisely, for $a \in G$, the map $\alpha_a : G \rightarrow G$ given by

$$\alpha_a(g) = a^{-1}g,$$

defines the particular permutation required to produce the routing table for PE a from the routing table at the identity PE. This can be achieved by a straightforward indirect addressing scheme indexing into the routing table for the identity PE.

In an analogous fashion, for an implicit scheme, we need only specify ρ_e (e is the identity PE), since, if we let $\rho_a(g) = \rho_e(a^{-1}g)$, then a message arriving at a, on its way to g, will be correctly forwarded. It is important to note that in this scheme, we require the ability to multiply and take inverses of elements in the group.

3.1.2 Labellings of Cayley Graphs

A key attribute of a Cayley graph is that it naturally inherits a labelling scheme from the underlying group: Vertices are labeled by the elements of the group while edges are labeled by elements of the generating set. An important task is to find an encoding of the graph (group elements and generators) that facilitates routing. One standard method of encoding the elements of a group is via a set of permutations. However, Jerrum [67] has shown that the general problem of determining optimal paths in a Cayley graph when the group elements and generators are encoded as permutations is PSPACE-complete. Clearly, the class of all Cayley graphs is too broad to expect the existence of a single optimal routing algorithm. Thus, we must focus our attention on specific classes of groups and specific generating sets.

Let H be a subgroup of G and suppose T is a right transversal for H in G. Every element, $g \in G$, may be expressed as a pair

$$h_g t,$$

where $h_g \in H$ and $t \in T$. In fact, this decomposition is unique. For, if g_1 and g_2 are in different right cosets, then obviously the representatives in T are different, and if both g_1 and g_2 reside in the same right coset, then $h_{g_1} \neq h_{g_2}$ unless $g_1 = g_2$. Now suppose we

have a tower of subgroups,

$$\{e\} = H_0 \leq H_1 \leq \cdots \leq H_n = G,$$

and T_i is a transversal for H_i in H_{i+1} $(0 \leq i \leq n-1)$. By proceeding inductively, we obtain a decomposition of the group elements into unique sequences of the form

$$[t_0, t_1, \cdots, t_{n-1}]$$

where $t_i \in T_i$ for $0 \leq i \leq n-1$. Although this decomposition is a convenient way to represent elements of a group, it may not be a simple matter to characterize the action[1] of a generator on an arbitrary tuple of this form, and thus efficiently determine a routing algorithm. (The edges of a Cayley graph are formed from this action.) Nevertheless, we have reduced the labelling problem to finding an "efficient" subgroup tower of the group and then determining appropriate transversals for the successive factors in the tower.

For the particular case in which G is a **solvable** group (see [89]), there exists a tower of subgroups

$$\{e\} = N_0 \lhd N_1 \lhd \cdots \lhd N_{k-1} = G$$

in which each successive quotient N_i/N_{i+1} is (isomorphic to) a *cyclic* group. By representing an element of a cyclic group of order n by an integer between 0 and $n-1$, we find a decomposition of any solvable group into sequences of integers, using a total of $O(\log|G|)$ bits. Consider, for example, the wreath product, $Z_2 \wr Z_n$ (recall from Chapter 2 that this is the underlying group of the butterfly graph and the cube-connected cycles graph). We have the following subgroup tower in this case (see [89]):

$$\{e\} \lhd Z_2 \lhd Z_2^2 \lhd \cdots \lhd Z_2^{n-1} \lhd Z_2^n \lhd Z_2 \wr Z_n = G.$$

By noting that $Z_2 \wr Z_n / Z_2^n \simeq Z_n$ and $Z_2^k / Z_2^{k-1} \simeq Z_2$, we elicit a decomposition of the elements of $Z_2 \wr Z_n$ into sequences of the form

$$[\beta_0, \beta_1, \cdots, \beta_{n-1}, \alpha],$$

[1] *i.e.*, the result of multiplication on the right.

where $\alpha \in Z_n$ and $\beta_i \in Z_2$, $1 \leq i \leq n$. In fact, this is the standard labelling (in reverse) used for both $\mathcal{B}_n$ and $\mathcal{C}_n$ (see Chapter 2). In the next section, we show how this labelling suggests a routing algorithm for these networks.

3.1.3 Routing Based on Coset Graphs

The labellings produced in the previous section suggest a recursive routing algorithm which uses routing in subgroups and (either left or right) quotient graphs. These algorithms arise out of constructive investigations of the diameter of Cayley graphs in terms of the diameter of coset graphs and Cayley graphs of subgroups.

Right coset graphs

The following diameter bound and subsequent routing scheme is similar to one presented in [10].

Proposition 4 *Let $\mathcal{G} = \Gamma(G, S)$ and suppose $H \leq G$. Then, the following diameter bound holds.*

$$\text{diam}(\mathcal{G}) \leq \text{diam}(\Gamma_R(G/H, S)) + \text{diam}_{\mathcal{G}}(H)$$

Proof. Any vertex of $\mathcal{G}$ can be reached by concatenating a path from the identity to some element $h \in H$ together with a path to the correct representative of H in G. A path between representatives is equivalent to a path in $\Gamma_R(G/H, S)$ (the edge labels are identical – see Theorem 1). □

Routing Scheme I. Let H be generated by a subset $S' \subset S$ and fix a right transversal T for H in G. Suppose we have routing algorithms for $\Gamma(H, S')$ and $\Gamma_R(G/H, S)$. Then, we provide each $g = h_g t \in G$ (where $h_g \in G$ and $t \in T$) with a label consisting of $\mathsf{label}(h_g)$ concatenated with $\mathsf{label}(Ht)$, denoted by $[\mathsf{label}(h_g), \mathsf{label}(Ht)]$. Routing in $\mathcal{G}$ is now performed as follows:

- **Step 1.** Use the routing algorithm for $\Gamma(H, S)$ to route to [label(h_g), label(H)].
- **Step 2.** Use the routing algorithm for $\Gamma_R(G/H, S)$ to route from [label(h_g), label(H)] to [label(h_g), label(Ht)].

For example, this reduces routing in $\mathcal{B}_n$ to routing in a cycle and routing in $\mathcal{D}_n$. Similarly, routing in $\mathcal{C}_n$ is reduced to routing in a cycle and routing in $\mathcal{S}_n$. The following result proves the efficiency of this routing method for a certain class of Cayley graphs including the previous two examples.

Proposition 5 *Let H and K be subgroups of G such that $K \cap H = \{e\}$. Then*

$$\text{diam}(\mathcal{G}) \geq \frac{1}{2}(\text{diam}(\Gamma_R(G/H, S)) + \text{diam}(\Gamma_R(G/K, S)))$$

Proof. In Chapter 5, we prove that $\mathcal{G}$ has a bijective embedding in $\Gamma_R(G/H, S) \times \Gamma_R(G/K, S)$ with dilation at most 2 (see Theorem 14). We now show that if (α, ρ) is a bijective embedding of $\mathcal{G}$ in $\mathcal{G}_1 \times \mathcal{G}_2$ with dilation d, then

$$\text{diam}(\mathcal{G}) \geq 1/d(\text{diam}(\mathcal{G}_1) + \text{diam}(\mathcal{G}_2)),$$

proving the proposition. Suppose, on the contrary, that $\text{diam}(\mathcal{G}) < 1/d(\text{diam}(\mathcal{G}_1) + \text{diam}(\mathcal{G}_2))$. Let u and v be two vertices in $\mathcal{G}_1 \times \mathcal{G}_2$ that are maximal distance apart from each other (*i.e.*, their distance apart is $\text{diam}(\mathcal{G}_1)+\text{diam}(\mathcal{G}_2)$), and consider a shortest path between $\alpha^{-1}(u)$ and $\alpha^{-1}(v)$ in $\mathcal{G}$. By assumption, the distance between these two vertices is less than $1/d(\text{diam}(\mathcal{G}_1)+\text{diam}(\mathcal{G}_2))$. But, because the dilation of (α, ρ) is d, this implies that there is a path between u and v in $\mathcal{G}_1 \times \mathcal{G}_2$ of length less than $\text{diam}(\mathcal{G}_1) + \text{diam}(\mathcal{G}_2)$, contradicting the choice of u and v. □

Left coset graphs

A complementary routing scheme can be developed based on left coset graphs, assuming that the underlying group is equipped with a quasi-minimal generating set. The scheme

uses routing in left coset graphs and routing in Cayley graphs of subgroups as subroutines.

Proposition 6 *Let $\mathcal{G} = \Gamma(G, S)$ be a quasi-minimal Cayley graph and suppose H is the subgroup of G generated by all elements of S except the last in the quasi-minimal ordering of S, denoted by S_H. Suppose also that K is the subgroup of H defined in the structure theorem for left coset graphs (Theorem 2). Then, the following diameter bound holds.*

$$\mathrm{diam}(\mathcal{G}) \leq \mathrm{diam}(\Gamma_L(G/H, S))(1 + \mathrm{diam}(\Gamma_L(H/K, S_H))) + \mathrm{diam}_G(H)$$

Proof. Given any path in $\Gamma_L(G/H, S)$ we can "lift" this to a path in $\mathcal{G}$ as follows. Observe that a single edge in $\Gamma_L(G/H, S)$ is equivalent to two left cosets of H in G that are adjacent in $\mathcal{G}$. Passage from one left coset of H in G to another (in $\mathcal{G}$), we must find a path in the first left coset to the appropriate left coset of K in H (*i.e.*, to one of the vertices in the appropriate bundle). This is equivalent to finding a path in $\Gamma_R(H/K, S)$. Now, in order to find a path in $\mathcal{G}$ between e and an arbitrary $g \in G$, we first find a path to the correct left coset of H in G (*i.e.*, the coset gH) by lifting an appropriate path in $\Gamma_L(G/H, S)$, followed by a path between two elements of the single left coset gH (recall that the induced subgraphs on gH and H are isomorphic). □

Routing Scheme II. Suppose we are given routing schemes for $\Gamma_L(G/H, S)$, $\Gamma_L(H/K, S_H)$ and $\Gamma(H, S_H)$. Order the bundles in each left coset of H in G so that the i-th bundle, denoted by b_i, corresponds to the i-th port of $\Gamma_L(G/H, S)$, denoted by $\mathsf{port}(i)$. We label $\mathcal{G}$ by concatenating the labels for $\Gamma(H, S_H)$ with the labels for $\Gamma_L(G/H, S)$ according to the decomposition of elements of G into unique products of the form uh_g, where u is a representative of the left coset uH and $h_g \in H$. To route from e to uh_g, we first route to uh' (for some $h' \in H$) using the routing algorithm for $\Gamma_L(G/H, S)$, followed by routing from uh' to uh, using the routing algorithm for $\Gamma(H, S_H)$. Initially, let $\mathsf{port}(i)$ denote the first port supplied by the routing algorithm for $\Gamma_L(G/H, S)$.

- **Step 1.** Route from the current bundle to bundle b_i, using the routing algorithm for $\Gamma_L(H/K, S_H)$.
- **Step 2.** Route across the s (or possibly s^{-1}) edge according to the routing algorithm for $\Gamma_L(G/H, S)$.
- **Step 3.** If the packet has arrived at the destination left coset then route to the final destination using the routing algorithm for $\Gamma(H, S_H)$; otherwise, let port(i) denote the next step in the routing algorithm for $\Gamma_L(G/H, S)$. Repeat Step 1.

As an example, this routing scheme shows how we may route in the butterfly network based on routing in ring networks and the hypercube (*cf.* Chapter 4). We remark that the routes produced may not be optimal since there are multiple ways to "lift" a path in $\Gamma_L(G/H, S)$ to a path in $\mathcal{G}$.

3.2 A General Strategy for Off-Line Permutation Routing

In this section, we present a general strategy for finding efficient *permutation routes* in parallel networks. The type of routes that we are most interested in finding are ones which have small dynamic congestion, *i.e.*, during each communication step taken to realize the route, the paths produced are vertex-disjoint. This is an important practical requirement, as its fulfillment obviates the need for costly queueing mechanisms and ensures the routes will complete with minimal delay. (If there is no congestion then the time to route is completely determined by the length of the longest path that any packet must take to reach its destination.)

The strategy applies to our benchmark networks, including mesh networks, hypercube networks, hypercube-derivative networks and a large class of double-ring networks. Also, we extend the strategy to several classes of Cayley networks, such as star networks and pancake networks, by exploiting their underlying group structure. Because of the

inherent symmetry of Cayley networks, we conjecture that they all support efficient off-line permutation routing. Our strongest general result proves that for every *subgroup tower* of a group G, there exists a *strong generating set* for G such that routing can be performed on the resultant Cayley graph in a number of steps which is at most four times the length of the subgroup tower.

In general, the routes produced by the strategy take a number of routing steps that is within a small constant factor of the diameter of the network. The starting point of our approach is derived from an algorithm that finds (in polynomial time) efficient permutation routes for the product network, $G \times H$, given efficient permutation routes for G and H. We also investigate the use of this algorithm for routing *multiple permutations*.

3.2.1 Background and Results

Efficient randomized algorithms for the permutation routing problem are known for many popular parallel networks such as hypercube networks [105], mesh networks [71] and butterfly networks [87]. However, much less is known about the characteristics that are necessary for an underlying topology to support efficient permutation routing. Having such knowledge about the structure of networks is clearly desirable. Firstly, it would simplify the task of designing routing algorithms for new networks, and secondly, it would increase our *fundamental* knowledge of the problem by clarifying the key features of a network that endow it with efficient routing capabilities. We approach the problem of off-line[2] permutation routing from this vantage point; our main aim is to develop a unified framework for finding efficient off-line routes on a wide range of topologies. In general terms, the main conclusion of this section states that a parallel network supports efficient[3] permutation routing if it has an "approximate" product structure or is derived in a uniform manner from such a structure. As special cases of the method, we find efficient,

[2]We assume global knowledge of the permutation while finding the routes.

[3]In the future, a set of routes will be called efficient if the number of links traversed is within a small constant factor of the optimal number.

Network	Size	Degree	Diameter	Routing
order-n mesh	n^2	≤ 4	$2n-2$	$3n-3$
order-n k-ary cube	k^n	n	$n\lfloor \frac{k}{2} \rfloor$	$(2n-1)(k-1)$
order-n butterfly	$n2^n$	4	$\lfloor \frac{3n}{2} \rfloor$	$4n-2$
order-n deBruijn	2^n	≤ 4	n	$2n-1$
order-n star	$n!$	$n-1$	$\lfloor \frac{3}{2}(n-1) \rfloor$	$4n-7$
order-(n,m) product-shuffle	2^{n+m}	≤ 8	$n+m$	$4n+2m-3$

Table 3.1: Comparison of routing times and diameters.

congestion-free, off-line routes for all of our benchmark networks defined in Section 2, in addition to the following:

- **high bandwidth networks**: for arbitrary positive integers $n, m \geq 2$ there exists a bit-serial network of size n^m and degree $(n-1)m$ which can route a permutation of n^m messages, each of nm bits in length, in $4(m-1)+2$ bit steps.
- **other networks**: these include the **star** networks [4], the **pancake** networks [4,48], a large class of **ring** networks, and the recently discovered **product-shuffle** [93] networks.

All of the routes arise from a single basic strategy in which the key step is to find a decomposition of a network into smaller subnetworks that are known to support efficient permutation routing. Although the decomposition and the techniques we use for finding it vary considerably between families, there is a single underlying approach.

The Importance of Off-Line Routing

The problem of off-line routing in parallel networks is of interest for several reasons: Firstly, by studying off-line routing we can establish the *existence* of efficient permutation routes. This may form a precursor to finding other types of routing algorithms.

Secondly, there are many application areas in which off-line routing is practical and,

in some cases, crucial. In particular, when the permutation to be routed is known at compile-time, off-line routing becomes applicable. For a program that is often executed, the initial cost of finding the most efficient routes (using off-line techniques) is offset by the savings accrued due to a shorter execution time. This situation arises in programming linear-algebra problems such as matrix multiplication and in SIMD routing applications [59]. In the latter example, off-line routing is particularly important if there are no queues available at the nodes making randomized or adaptive on-line algorithms difficult to implement. An example of a parallel computer using this approach to routing is the IBM GF-11 which has a topology of a Beneš network (see [5]).

Thirdly, the problem of emulating one network on another often reduces to the problem of routing an appropriate permutation on the target network (*e.g.*, see [8]). In this case, finding the paths that route the required permutation represents a one-time cost.

Finally, finding off-line routes can explain *why* good routings exist, exposing important underlying structural properties of the network.

For the remainder of this chapter, we employ the following terminology: The term **routes** refers to the *paths* produced by an off-line algorithm for routing a permutation. The phrase **routing algorithm** refers to the *algorithm* that produces these paths. Most of the routes we find are *congestion-free*, so that no queues are necessary at the PEs of the network.

Summary of Results and Outline of the Approach

Instead of attempting to find algorithms that outperform the best-known ones (although most of the routes that we find are close to optimal) we concentrate on developing a general methodology with which to approach the problem for many varying topologies. We mention here that there is a large body of literature on off-line routing on multistage interconnection networks (such as the Omega network [82], extra stage cube network [1],

and Gamma network [83]), although many of these networks have very similar topologies (indeed, many of them have been shown to be isomorphic [111]) and the routing algorithms that have been developed often apply specifically to a single class of topologies.

The focus of our investigation is derived from the following two general observations:

1. **Recursive structure.** Many networks have a recursive structure, being composed of smaller and simpler networks. For example, both the mesh network and the hypercube network are constructed via the direct product of cycles.

2. **Group-theoretic structure.** Following our remarks in Chapter 2, many networks can be expressed as Cayley graphs.

In regard to the first observation, we show that if the networks G and H support efficient routing, then so does the direct product network $G \times H$. There is no additional congestion in the routes for $G \times H$ over and above that incurred by the routes for G and H. We call the algorithm that produces routes for $G \times H$ the *direct product routing algorithm* and note that it is similar to one developed by Slepian [98] in the context of a class of non-blocking switching networks known as three-stage Clos networks (also, see [21,39]).

By interpreting the direct product routing algorithm in a very general light, we propose a strategy for finding efficient routes in a parallel network and provide several examples showing that the strategy is sufficient to find routes for many familiar topologies. We also investigate the idea of using the direct product routing algorithm to route *multiple permutations* on networks. For example, we show how to route n independent permutations *simultaneously* on an n-dimensional hypercube in $2n - 1$ steps (assuming a *multi-port* model of communication) and then apply this to obtain a fast off-line bit-serial routing algorithm for the hypercube. (This is related to results by Greenberg and Bhatt [53] and Aiello et al [2].)

Following the second observation, we present a general investigation of ways in which

the direct product routing algorithm can be extended to Cayley networks. These results indicate that many Cayley graphs are capable of efficient routing. More specifically, we present three theorems for routing on Cayley graphs. The first theorem applies to routing on Cayley graphs of *abelian* groups and semidirect products of cyclic groups. It generalizes to an algorithm for routing on the *star* and *pancake* networks: We show how to route a permutation on these networks in a near-optimal number of steps. Finally, we prove that given a group G and a subgroup tower for G, there exists a strong generating set S, for which routing can be performed on $\Gamma(G, S)$ in a number of steps which is at most 4 times the length of the subgroup tower.

In summary, this section is organized as follows. Section 3.2 describes the general routing strategy (Sections 3.2.1 and 3.2.2), provides several examples (Sections 3.2.3 and 3.2.4), extends the routing strategy to include routing multiple permutations (Section 3.2.5) and presents a useful graph-theoretic extension of the direct product routing algorithm (Section 3.2.6). Section 3.3 extends the techniques of Section 3.2 to several families of Cayley graphs.

3.2.2 The General Routing Strategy

Let $\mathcal{G}$ and $\mathcal{H}$ be networks that each come equipped with a routing algorithm. Form the product network, $\mathcal{F} = \mathcal{G} \times \mathcal{H}$. Our aim is to design a routing algorithm for the new network $\mathcal{F}$ by using the routing algorithms for the factors as *subroutines*. We apply the following three-phase regimen.

1. Route some set of permutations (in parallel) on the copies of $\mathcal{G}$.

2. Route some set of permutations (in parallel) on the copies of $\mathcal{H}$.

3. Route some set of permutations (in parallel) on the copies of $\mathcal{G}$.

Alternatively, we could apply the three-phase regimen by first routing on $\mathcal{H}$, followed by $\mathcal{G}$, followed by $\mathcal{H}$. Let $T(\mathcal{N})$ denote the maximum number of steps (equivalently, this is the length of the longest path) required to complete a permutation route on an arbitrary network $\mathcal{N}$. If the routes for the factor graphs are optimal[4] then our algorithm produces routes which are at worst an additive factor of $\min\{T(\mathcal{G}), T(\mathcal{H})\}$ away from the optimal algorithm for $\mathcal{F}$ since the diameter of a direct product is just the sum of the diameters of the factors.

The Routing Algorithm for $\mathcal{F} = \mathcal{G} \times \mathcal{H}$

Let ρ be any permutation of the network $\mathcal{F}$. Consider the naive method for routing ρ on $\mathcal{F}$:

1. Route each packet to the correct row.
2. Route each packet to the correct column.

This fails to be a congestion-free algorithm because there may be several packets in some column that have their destination in a single row (causing congestion at the intersection of the column and row.) By using an initial extra phase we can resolve this problem. We first "rearrange" each column so that, after the rearrangement, each row consists of a set of packets whose destinations are all in *distinct* columns. In this case, a *permutation* of each row will then be required to get each packet to its correct column after the rearrangement. Once all of the packets are in their correct columns, a final permutation of each column will suffice to get each packet to its correct destination. This final phase is indeed a permutation since all destinations of packets are distinct. Thus, the aim of the first phase of the algorithm can be summed up as follows:

[4] *i.e.*, they complete in the smallest possible number of steps.

For each row R, find a set of packets P_R, one per column, such that each packet in P_R has its destination in a distinct column.

To this end, construct a bipartite multigraph $\mathcal{B} = (C \cup C', E)$ as follows. Let C and C' each represent the set of columns of $\mathcal{F}$. There is an edge between c_i and c'_j for each packet in column i whose destination is in column j. Since ρ is a permutation, it follows that $\mathcal{B}$ is a regular bipartite multigraph. Thus, $\mathcal{B}$ can be decomposed into a set of disjoint perfect matchings (see [31]). Note that the packets involved in a single perfect matching all have their destinations in distinct columns. Hence, for each row R, we use all of the packets that correspond to a single perfect matching for our set P_R. Each of these sets, P_R, (for each row R) is "lifted" to row R during the first phase of the algorithm. Since each packet is involved in precisely one perfect matching, the mapping of packets in each column during the first phase of the algorithm is a permutation of the column.

We summarize the above discussion in the following theorem.

Theorem 4 *Given routing algorithms for networks $\mathcal{G}$ and $\mathcal{H}$, there is a routing algorithm for the product network $\mathcal{G} \times \mathcal{H}$. The routes produced by this algorithm take at most*

$$T(\mathcal{G}) + T(\mathcal{H}) + \min\{T(\mathcal{G}), T(\mathcal{H})\}$$

steps to complete on $\mathcal{G} \times \mathcal{H}$, where $T(\mathcal{G})$ (resp., $T(\mathcal{H})$) represents the number of steps to complete a permutation route on $\mathcal{G}$ (resp., $\mathcal{H}$). Furthermore, the routing algorithm runs in deterministic polynomial-time (in the size of $\mathcal{F}$) and no additional congestion is accrued above that caused by the routing algorithms for $\mathcal{G}$ and $\mathcal{H}$.

Proof. The proof of the first assertion follows directly from the above discussion. The timing of the routing algorithm, *i.e.*, the time complexity of computing the routes, is dominated by the time taken to decompose the graph $\mathcal{B}$ into perfect matchings, a problem

for which there are several sequential polynomial time (see [85]) as well as randomized polylogarithmic time (see [77]) parallel solutions. □

A General Routing Strategy

There is a useful alternative way of viewing the proof of the direct product routing algorithm:

Let S be the cartesian product of two sets S_1 and S_2. Given any permutation ρ of S, there is a factorization of ρ as a product of the following form:

$$\rho = \delta\sigma\delta'$$

where σ fixes the first component of each element in S and both δ and δ' fix the second component of each element of S. The reader may easily verify that this is an equivalent formulation of Theorem 4. It suggests the following widely applicable divide-and-conquer strategy to designing an off-line permutation routing algorithm for a given network $\mathcal{N}$.

- **Step 1.** Partition the vertex set of $\mathcal{N}$ into equal-sized disjoint *blocks*.
- **Step 2.** Find a routing algorithm for routing permutations of the blocks.
- **Step 3.** Find a routing algorithm for routing permutations "between the blocks".

Since any permutation of $\mathcal{N}$ can be factorized into $\delta\sigma\delta'$ where δ and δ' map packets within a fixed block to the same block and σ maps packets between the blocks, the above strategy in conjunction with Theorem 4 suffices to route any permutation of $\mathcal{N}$. There are two obvious questions that arise in regard to this strategy: Firstly, how do we choose the blocks efficiently and secondly, how do we route between the blocks? The remainder of this chapter is devoted to answering these questions for several families of graphs, including our benchmark topologies.

Applications to Specific Networks

We first apply Theorem 4 directly to some common parallel networks.

Mesh networks. For a set of n processors connected in a line, $\mathcal{L}_n$, permutation routing can be performed in $n-1$ steps (since the links are bidirectional). Consider the $n \times n$ **mesh** network, $\mathcal{M}_n = \mathcal{L}_n \times \mathcal{L}_n$. Theorem 4 yields $T(\mathcal{M}_n) = 3n-3$. We remark that this result appears to be the best off-line, congestion-free algorithm currently known for this network. Results of Nassimi and Sahni [78] show that more efficient routing can be performed on the mesh for a proper *subclass* of permutations, while results of Kunde [72] show how to route deterministically on the mesh in $2n+O(n/f(n))$ steps with congestion $f(n)$. In this latter case, the low-order terms imply a routing time of more than $3n-3$ steps when $f(n)=1$.

Cube networks. Since $\mathcal{H}_n$ is isomorphic to the direct product of n copies of the complete graph on two vertices, by applying the routing algorithm inductively we obtain the recurrence

$$T(\mathcal{H}_n) = T(\mathcal{H}_{n-1}) + 2$$

for $n \geq 2$. Since $T(\mathcal{H}_2) = 1$, we get $T(\mathcal{H}_n) = 2n-1$. In general, routing on an n-dimensional k-ary cube can be done in time $(2n-1)(k-1)$.

Beneš networks. The routing algorithm for $\mathcal{H}_n$ that our formulation leads to is equivalent to the well-known congestion-free routing scheme of **Beneš** [21]. Also, since the bipartite graph, $\mathcal{B}$, of the direct product routing algorithm has degree two in this case, there is an efficient PRAM algorithm for finding the required matchings and hence an efficient parallel distributed algorithm for routing a given permutation (*cf.* the technique in [79]).

deBruijn networks. The previous example yields a *normal* algorithm (in the sense of Ullman [103]) and hence can be executed on the deBruijn graph without loss of efficiency.

Product-Shuffle networks. Rosenberg [93] defines the **product-shuffle** network, $\mathcal{PS}_{n,m}$, as the direct product $\mathcal{D}_n \times \mathcal{D}_m$. It is shown to be superior to both deBruijn networks and butterfly networks in several respects. By the previous example, routing can be performed on $\mathcal{D}_n$ in $2n-1$ steps. Thus, if $n \leq m$, then Theorem 4 implies that there is a routing algorithm for $\mathcal{PS}_{n,m}$ that takes $4n+2m-3$ steps, which is within a small constant factor of the diameter of $\mathcal{PS}_{n,m}$.

Routing on Butterfly Networks

In this subsection we show how to route a *full* permutation of the nodes of the butterfly network, $\mathcal{B}_n$, in $O(n)$ steps without congestion. To attain our goal, we shall take advantage of the levelled structure of the butterfly. Let ρ be an arbitrary permutation of $\mathcal{B}(n)$. By appealing to the method of the general routing strategy, we can factorize ρ as $\delta\sigma\delta'$ where δ and δ' fix the columns setwise[5] and σ fixes the levels setwise. We deal with each of these types of permutations in turn:

- Realizing any permutation that fixes the columns setwise takes $n-1$ steps since each column of the n-th order butterfly is an n-vertex cycle.
- We achieve an arbitrary permutation of each level of the butterfly (routing "between the blocks") by using the Beneš routing technique. Computing the paths to effect a permutation of the i-th level ($0 \leq i \leq n-1$) is equivalent to applying the direct product routing algorithm inductively to the hypercube $\mathcal{H}_n$ starting from the $(i+1)$-st dimension. (Normally, the Beneš routing technique starts by routing in the first dimension.) There are two phases in routing a single level: First, the packets pass from level to level in increasing order (finishing at their original level). In the second phase, the packets pass from level to level in decreasing order until they return to their original level. This two phase process takes $2n$ steps. Note that

[5]That is, packets in each column have their destinations in the same column.

we can perform the routing for *all* of the levels in parallel without any congestion (by pipelining) in the *same* number of steps.

Hence, the total number of steps taken to complete a full permutation route of a butterfly network is $(n-1)+2n+(n-1) = 4n-2 = O(n)$. We also note that since the order-n cube-connected cycles network can simulate the order-n butterfly network with a slowdown factor of 2, routing can be performed on the cube-connected cycles in $6n-2$ steps.

Routing Multiple Permutations

Observe that, under the multi-port model of communication, two independent permutations can be routed on $\mathcal{G} \times \mathcal{H}$ *simultaneously* using the direct product routing algorithm. (Use the $\mathcal{G}$, $\mathcal{H}$, $\mathcal{G}$ regimen for the one permutation and the $\mathcal{H}$, $\mathcal{G}$, $\mathcal{H}$ regimen for the other.) To avoid congestion, during each phase we must wait for the route that takes the longest time to complete before proceeding with the next phase. Thus, any two permutations can be routed on $\mathcal{G} \times \mathcal{H}$ in $3\max\{T(\mathcal{G}_1), T(\mathcal{G}_2)\}$ steps. Note that we are now taking full advantage of all of the communication ports at each processor (*cf.* [53]). It follows by a simple induction that we can route n simultaneous permutations on an iterated direct product $\mathcal{G}_1 \times \mathcal{G}_2 \times \ldots \mathcal{G}_n$ in $(2n-1)\max_i\{T(\mathcal{G}_i)\}$ steps.

Applying this observation to the n-dimensional hypercube, we see that we can route any n permutations simultaneously in $2n-1$ steps. This observation can also be applied to the problem of **bit-serial** routing in a hypercube. If the bandwidth of each wire in the network is a single bit and our packets are M bits long, then sending a single packet from one processor to an adjacent one requires $\Omega(M)$ time steps. By allowing a message to spread itself out in the network (using the wormhole routing technique [42], for example), Aiello et al [2] have developed a randomized algorithm for routing a permutation in this model in $O(M+n)$ bit steps. In a somewhat related setting, Greenberg and Bhatt [53] used a "multiple copy" embedding of a cube-connected cycles network in the hypercube

to route n permutations, one on each copy of a cube-connected cycles network, in time $O(M)$. By appealing to the direct product routing algorithm, we obtain a similar result:

Theorem 5 *Suppose $\mathcal{H}_n$ is a bit-serial hypercube and that the packets at each node are M bits long. Then, any permutation can be routed in $\lceil \frac{M}{n} \rceil (2n-1)$ bit steps without congestion.*

Proof. Split each packet into $\lceil \frac{M}{n} \rceil$ pieces each n bits long (except for maybe the last piece). Now route each piece using the observation preceding this theorem. □

Finally, we present an example of how the direct product routing algorithm can be applied to high-bandwidth networks[6]. Let $\mathcal{K}_n$ denote a fully connected bit-serial network with n nodes. Suppose each packet consists of n bits. One can route any permutation of these "large" packets on $\mathcal{K}_n$ in 2 steps: First, each node sends, in parallel, the i-th bit of its packet to its i-th neighbor. Each node then collects the n bits of the packet that was destined to it in one more communication step. In general, by using the direct product routing algorithm, the direct product of m copies of $\mathcal{K}_n$ can route a permutation of n^m packets each of length nm in $4(m-1)+2$ steps. This represents an "extreme" situation: even though the size of the network increases by a multiplicative factor of n as we add copies of $\mathcal{K}_n$, the time to route an arbitrary permutation increases by only a *constant* additive factor.

A Graph-Theoretic Extension

We now derive a useful graph-theoretic extension of Theorem 4. Suppose $\mathcal{G}$ and $\mathcal{H}$ are graphs with $|\mathcal{G}| = n$ and $|\mathcal{H}| = m$. Let

$$\mathrm{p} = (\pi_1, \pi_2, \cdots, \pi_m)$$

[6]This example is due to D. Greenberg [52].

be an m-tuple of permutations of an n-element set. Define the **permuted product** graph with respect to $\mathbf{p}$, denoted $\mathcal{G} \times_{\mathbf{p}} \mathcal{H}$, to be the (standard) direct product of $\mathcal{G}$ and $\mathcal{H}$ except that the *vertices* in the i-th column are permuted by π_i. More formally, let each vertex h_i in H correspond to the i-th column. There is an edge between (g, h_i) and (g', h_j) (where g, g' are vertices of G and h_i, h_j are vertices of H) if either (g, g') is an edge of G and $h_i = h_j$, or, (h_i, h_j) is an edge of H and $g = \pi_j^{-1}(g')$.

Note that the edges in each column (*i.e.*, each copy of $\mathcal{G}$) remain fixed but the edges in the copies of $\mathcal{H}$ are permuted according to $\mathbf{p}$. The following theorem can be deduced in a straightforward manner from Theorem 4 and hence we omit the proof.

Theorem 6 *Let $\mathcal{F}$ be a permuted product of two graphs $\mathcal{G}$ and $\mathcal{H}$. If permutation routing can be performed on $\mathcal{G}$ and $\mathcal{H}$ in time $T(\mathcal{G})$ and $T(\mathcal{H})$, respectively, then permutation routing can be performed on $\mathcal{F}$ in time*

$$T(\mathcal{F}) = T(\mathcal{G}) + T(\mathcal{H}) + \min\{T(\mathcal{H}), T(\mathcal{G})\}.$$

Example. Define the **generalized Peterson graphs**, denoted by $\mathcal{GP}_{n,k}$, $n \geq 2$, $1 \leq k \leq n-1$, as follows. The vertex set is $\{u_0, u_1, \ldots, u_{n-1}, v_0, v_1, \ldots, v_{n-1}\}$ and the edge set is

$$\{(u_i, u_{i+1}), (u_i, v_i), (v_i, v_{i+k}) : i = 0, 1, \ldots, n-1\},$$

where all subscripts are reduced modulo n.

Let π_1 be the identity permutation on the set $\{1, 2, \ldots, n\}$ and π_2 be the permutation that maps i to $(i+k)$ mod n, for $1 \leq i \leq n$. If we let $\mathbf{p} = \{\pi_1, \pi_2\}$, then it is easily verified that $\mathcal{GP}_{n,k}$ is the permuted product $\mathcal{R}_2 \times_{\mathbf{p}} \mathcal{R}_n$. Hence, by applying Theorem 6, any permutation of this graph can be routed in $n+1$ communication steps.

3.2.3 Off-Line Routing in Cayley Graphs

We now turn to an exploration of the ability of a Cayley network to support efficient off-line permutation routing. The symmetry of Cayley graphs seems to play an important role in ensuring the existence of efficient congestion-free routes and although we are nowhere near a complete proof, we conjecture the following **Cayley Graph Routing Thesis.**

> Every family of Cayley graphs has an efficient (*i.e.*, of length O(diameter)) set of congestion-free permutation routes.

Since the problem of determining the diameter of Cayley graphs is far from resolved, a constructive proof of this conjecture seems somewhat optimistic. However, there are two natural ways to restrict this conjecture. Firstly, we may place a restriction on the group structure (such as restricting ourselves to abelian or solvable groups, for instance) and secondly, we may place a restriction on the generating set. At the end of this section, we follow the latter approach and prove that, for every group G, given a subgroup tower for G of length n, there exists a strong generating set, S, for which off-line permutation routing can be performed on the Cayley graph, $\Gamma(G,S)$, in $O(n)$ steps.

In the first part of this section, we prove two other results on Cayley graph routing. The first result is quite general, pertaining mainly to Cayley graphs of abelian groups and Cayley graphs of semidirect products of cyclic groups. If S generates G and H is a subgroup of G, the method utilizes disjoint copies of a spanning tree of $\Gamma_R(G/H,S)$ that lie in $\Gamma(G,S)$ for procuring an efficient set of routes for $\Gamma(G,S)$. The second result extends the routing algorithm to a family of Cayley graphs known as *star graphs*.

Using a Spanning Tree of $\Gamma_R(G/H,\Pi)$

The following theorem shows how we may view certain Cayley graphs as permuted product graphs.

Theorem 7 *Let $\Gamma(G,S)$ be a Cayley graph. Let N be a normal subgroup of G generated by a subset $S' \subset S$ and suppose routing can be performed on $\mathcal{N} = \Gamma(N,S')$ in time $T(\mathcal{N})$. Then, routing can be performed on $\Gamma(G,S)$ in time*

$$T(\mathcal{N}) + O\left(\frac{|G|}{|N|}\right)$$

Proof. Let $\mathcal{T}$ denote a spanning tree of $\Gamma_R(G/N,S)$. Partition the vertices of $\Gamma(G,S)$ into $|N|$ blocks, where each block is a copy of $\Gamma(N,S')$ (using Proposition 1). Since N is normal in G, $\mathcal{T}$ has exactly one vertex in each left coset and hence, we can use $|N|$ copies of $\mathcal{T}$ (by Theorem 1) as the means for routing between the blocks. It follows that there exists some vector of $|G|/|N|$ permutations, $\mathbf{p}$, for which $\Gamma(G,S)$ contains the permuted product $\Gamma(N,S) \times_{\mathbf{p}} \mathcal{T}$ as a subgraph. The result follows from Theorem 6 and the observation that routing on a tree $\mathcal{T}$ takes $O(|\mathcal{T}|)$ steps. □

Example. Theorem 7 can be applied to all abelian groups, Hamiltonian groups[7] and semidirect products of cyclic groups. Loosely speaking, in this case, the theorem yields a routing algorithm that can be viewed as a generalization of routing on mesh networks. (It applies effectively to a large class of double ring networks.) For definiteness, let A be an arbitrary abelian group with a quasi-minimal generating set, S. Suppose the quasi-minimal ordering is

$$s_1, s_2, \ldots, s_m,$$

and $o(s_i) = \ell_i$, for $1 \leq i \leq m$. Let A_i, $1 \leq i \leq m-1$, be the subgroup of A generated by the first i generators. It is easy to see that the diameter of $\Gamma(A,S)$ is

$$\sum_{i=1}^{m} \lfloor \frac{\ell_i}{2} \rfloor = \Omega(\sum_{i=1}^{m} \ell_i).$$

[7] A group is said to be Hamiltonian if all of its proper subgroups are normal.

Also, by the normality of A_{m-1}, the right coset graph $\Gamma_R(A_m/A_{m-1}, S)$ forms an ℓ_m-cycle. By applying Theorem 7 inductively, we obtain routes for $\Gamma(A, S)$ that take at most

$$O(\sum_{i=1}^{m} \ell_i - 1) = O(\sum_{i=1}^{m} \ell_i)$$

steps to complete.

Routing on Star Graphs

When the subgroup mentioned in Theorem 7 is not normal, the Cayley graph may not decompose into a permuted product. However, by a more careful analysis of the coset graphs involved, it may still be possible to adapt the direct product routing algorithm. In this section, we implement the strategy on the *star graphs*. Our technique also applies to the *pancake graphs* [48,4] and the *bubble-sort graphs* [4] although we shall only present the details for the star graphs.

Let Π_n be the set of all transpositions of the form $(1, i)$, $2 \leq i \leq n$. The group generated by Π_n is $\mathrm{Sym}(n)$, the symmetric group on n points. Define the order-n **star graph**, denoted by $\mathcal{X}(n)$, to be the Cayley graph $\Gamma(\mathrm{Sym}(n), \Pi_n)$. Define an n-**spoke graph**, denoted by $\mathcal{S}(n)$, to be an n-vertex graph that has a single **center** vertex, $n-1$ **spoke** vertices and an edge between the center vertex and each spoke vertex.

Proposition 7 *The star graph $\mathcal{X}(n)$ has the following properties:*

- *$\mathcal{X}(n)$ can be decomposed into $(n-1)!$ vertex-disjoint n-spoke graphs.*
- *Let $\mathcal{S}(n)$ be a fixed n-spoke graph in the decomposition of $\mathcal{X}(n)$ just mentioned. Then every left coset of $\mathrm{Sym}(n-1)$ in $\mathrm{Sym}(n)$ either has a vertex connected to $\mathcal{S}(n)$ via a $(1, n)$ edge or a vertex that is in $\mathcal{S}(n)$.*

Proof. Consider the right coset graph, $\Gamma_R(\mathrm{Sym}(n)/\mathrm{Sym}(n-1), \Pi_n)$. Each right coset is defined by the image of the point n. For example, if n is fixed then we get the subgroup

$\mathrm{Sym}(n-1)$. Now, by simple calculation, it can be shown that the right coset graph is an n-spoke graph with $n-2$ self-loops on each of the spokes. Thus, by Proposition 2, $\mathcal{X}(n)$ can be decomposed into $(n-1)!$ disjoint n-spoke graphs.

For the second assertion of the proposition, note that each *left* coset of $\mathrm{Sym}(n-1)$ in $\mathrm{Sym}(n)$ is defined by the preimage of the point n. Let $\mathcal{S}(n)$ be a fixed n-spoke graph and suppose the center vertex of $\mathcal{S}(n)$ is the permutation τ. Then the spokes of $\mathcal{S}(n)$ are the permutations $\tau \cdot (1,j)$, for $2 \leq j \leq n$. Now consider the set of permutations

$$A = \{\tau \cdot (1,j) \cdot (1,n) \ : \ 2 \leq j \leq n\}.$$

For each of these permutations, the preimage of n is the preimage of j under τ. Also, since τ is a permutation, the preimage of each point j is distinct. Hence, each element of A is in a distinct left coset. □

We refer the reader to Figure 2.2 at the end of Chapter 2 for an example of this decomposition for $\mathcal{X}_3$.

Theorem 8 *$\mathcal{X}(n)$ can perform congestion-free routing in $O(\log N / \log\log N)$ steps, where $N = n!$.*

Proof. The method employs a divide and conquer strategy on the *left* cosets of $\mathcal{X}(n)$. The idea mirrors the proof of Theorem 7. We reduce routing on $\mathcal{X}(n)$ to n parallel routes on $\mathcal{X}(n-1)$ (which act as the blocks). We use the right coset graph $\Gamma_R(\mathrm{Sym}(n)/\mathrm{Sym}(n-1), \Pi_n)$ to route between the blocks. By Proposition 3, we can use the spoke graphs (*i.e.*, the copies of $\Gamma_R(\mathrm{Sym}(n)/\mathrm{Sym}(n-1), \Pi_n)$ as *partial* "rows" since each copy of $\mathcal{X}(n-1)$ has a vertex that is either in a given spoke graph or a vertex that is adjacent to it. Thus, in at most 2 steps, a message in some spoke graph can be routed to any given copy of $\mathcal{X}(n-1)$ as follows:

Route to the spoke vertex that is adjacent to the appropriate left coset and

then follow the $(1, n)$ edge. If the left coset involved is the one which fixes the point n, then only the first step is required.

Notice that the above routing can be performed in parallel on *all* of the spoke graphs since they are edge-disjoint and each vertex has exactly one $(1, n)$ edge attached to it. Hence, by induction, the total time for the routing to complete is

$$T(\mathcal{X}_n) = T(\mathcal{X}_{n-1}) + 4,$$

which evaluates to $4n - 7 = O(\log N / \log \log N)$, since $T(2) = 1$. □

Routing in Cayley Graphs with Strong Generating Sets

Let G be a group and suppose we have a tower of subgroups of length n,

$$\{e\} = H_0 \leq H_1 \leq \cdots \leq H_n = G.$$

Suppose further that U_i is a right transversal for H_i in H_{i+1} $(0 \leq i \leq n-1)$. Thus, each element of G can be uniquely expressed as a product of the form

$$u_0 u_1 \cdots u_{n-1},$$

where each $u_i \in U_i$. We define a **strong generating set**, S, for G to be the union of the transversals U_i. That is,

$$S = \bigcup_{i=0}^{n-1} U_i.$$

Strong generating sets play a central role in computational group theory and the graph isomorphism problem (see, for example, [47]). Also, Cayley graphs equipped with strong generating sets were studied by Blaha [28] in the context of on-line routing and connectivity. Our aim is to define a particular type of strong generating set for which the resultant Cayley graph supports off-line permutation routing. We require the following fact which can be found on page 12 of [114]. We include a proof for the sake of completeness.

Lemma 1 *For any subgroup H of a group G, there exists a transversal for H in G that is simultaneously a left transversal and a right transversal.*

Proof. Define a bipartite graph, $\mathcal{B} = (L \cup R, E)$, where L represents the set of left cosets of H in G and R represents the set of right cosets of H in G. There is an edge between $l \in L$ and $r \in R$ whenever there is an element of the group simultaneously in the left coset represented by l and the right coset represented by r. Notice that a transversal of the desired type corresponds to the existence of a perfect matching in $\mathcal{B}$. By Hall's marriage theorem (see, for example, [88]), there is a perfect matching in $\mathcal{B}$ if and only if for any subset, $A \subseteq L$, of size k, the neighborhood set of A is at least k. This is indeed the case, since the number of edges emanating from A is equal to $|H|k$ and each vertex in R has exactly $|H|$ edges associated with it. □

Now let S be a strong generating set for G, where each transversal in the given subgroup tower is simultaneously a left transversal and a right transversal. Our technique follows the technique used for finding efficient routes for the star graphs (see the previous section).

Consider the right coset graph, $\Gamma_R(G/H_{n-1}, S)$. We will use copies of this right coset graph as "columns" and copies of induced subgraphs of left cosets of H_{n-1} in H_n as "rows". Recall that the central obstacle to reducing the problem to routing in induced subgraphs of left cosets is that a mechanism for routing between the left cosets may not be at all evident. However, for our particular choice of S, this is not the case. Define $k = |G : H_{n-1}|$. The following lemma exposes the reason for our choice of S.

Lemma 2 *$\Gamma(G, S)$ contains $|H_{n-1}|$ disjoint copies of a k-spoke graph with the following 2 properties:*

- *The set of vertices involved in each spoke graph is a left transversal of H_{n-1} in G.*

- *Every vertex of $\Gamma(G,S)$ is in a unique k-spoke graph.*

Proof. Observe that $\Gamma_R(G/H_{n-1},S)$ is a k-spoke graph and that the copy of this graph in $\Gamma(G,S)$ that includes e as one of its vertices has the first property stated in the lemma. By Theorem 1, there are $|H_{n-1}|$ disjoint copies of $\Gamma_R(G/H_{n-1},S)$ in $\Gamma(G,S)$, since $\Gamma_R(G/H_{n-1},S)$ is a tree. Recall that each copy of $\Gamma_R(G/H_{n-1},S)$ in $\Gamma(G,S)$ may be obtained by left-multiplying the copy that includes e by some element in G. Hence, since left multiplication of a left transversal by an element of G is again a left transversal, the result follows. □

The routing method now proceeds along the same lines as in the previous section. In effect, we have reduced routing in $\Gamma(G,S)$ to routing in $\Gamma_R(G/H_{n-1},S)$ and the induced subgraphs on the left cosets of H_{n-1} in G.

Assuming the multi-port model of communication, the total time to route an arbitrary permutation with this method is given by the solution to the recurrence, $T(n) = 4 + T(n-1)$, $T(1) = 1$. The reason that $T(1) = 1$ is because a transversal of $\{e\}$ in H_1 gives rise to a complete graph on $|H_1|$ vertices. Hence, the total time to route an arbitrary permutation is $4n-3$ steps. We summarize this result in the following theorem.

Theorem 9 *For every group G and subgroup tower of G of length n, there is a strong generating set S, for which off-line permutation routing can be performed on $\Gamma(G,S)$ in at most $4n-3$ steps assuming the multi-port model of communication.*

CHAPTER 4

Routing in Faulty Networks

This chapter presents two primary results on the topic of fault-tolerant routing in parallel networks. We adopt a fault model which assumes that if a PE fails then messages can no longer be routed through it, *i.e.*, all communication links incident to that PE are unusable. We are interested in the following two versions of the reliability problem.

A: Fault-tolerant permutation routing. In Section 4.2, we modify the direct product routing algorithm presented in the previous chapter to route permutations in a network under the assumption that some of its PEs are inoperable. For a large family of networks, our results show that permutations among the remaining live PEs can be routed without *any* loss in efficiency.

B: Fault-tolerant point-to-point routing. When can we ensure the existence of fault-free communication paths between pairs of (non-faulty) PEs in a network assuming some of the network's PEs are faulty? In Section 4.2, we focus on Cayley networks, and derive a general inductive tool for analyzing their connectivity. As corollaries, we prove that all minimal Cayley graphs have optimal connectivity (first proved by Godsil [49]) and that any quasi-minimal Cayley graph in which all of the generators are of order 2 has optimal connectivity (this was a previous result of Akers and Krishnamurthy [3]). We illustrate our results by considering the wreathed networks (defined in Chapter 2) and generalized mesh networks. The idea behind the proof of the main theorem motivates the study of *quotient connectivity* for a network, a useful measure for studying a network's resilience to faults occuring in *clusters*.

4.1 Fault-Tolerant Permutation Routing

Let us assume that we are using the direct product routing algorithm to route permutations in a network. Suppose some subset of the PEs of the network fails.

Problem statement: Extend the direct product routing algorithm to route any permutation of the remaining live nodes.

We assume the notation developed in the proof of Theorem 4.

Theorem 10 *Let $\mathcal{G}$ be any network such that, in the presence of any p faults, $\mathcal{G}$ can route an arbitrary permutation among the remaining live nodes in time $T(\mathcal{G})$. Let $\mathcal{H}$ be any network such that, in the presence of a single fault, $\mathcal{H}$ can route an arbitrary permutation among the remaining live nodes in $T(\mathcal{H})$ steps. Form the product network $\mathcal{G} \times \mathcal{H}$ and suppose that a single copy of $\mathcal{G}$ in $\mathcal{G} \times \mathcal{H}$ contains p faults. Then, an arbitrary permutation of the live nodes of $\mathcal{G} \times \mathcal{H}$ can be performed in $T(\mathcal{H}) + 2T(\mathcal{G})$ steps.*

Proof. Let $|\mathcal{G}| = n$ and $|\mathcal{H}| = m$ (*i.e.*, there are n rows and m columns). We divide the proof into two cases.

A single fault. Initially, we assume the existence of a single fault in $\mathcal{G} \times \mathcal{H}$ located at column i and row j. Recall from Theorem 4 the definition of the bipartite multigraph $\mathcal{B}$. The key problem arises when we attempt to find a perfect matching of $\mathcal{B}$ to route on the j-th row. Since there is a fault in this row we require a perfect matching among only $m-1$ vertices (recall that each vertex of $\mathcal{B}$ represents a distinct column) of $\mathcal{G} \times \mathcal{H}$. More precisely, we require the existence of a perfect matching in the induced subgraph of $\mathcal{B}$ formed by all of the vertices except c_i and c'_i. The following lemma proves the existence of a perfect matching in graphs of this type.

Lemma 3 *Let $\mathcal{Y} = (L \cup R, E)$ be a bipartite multigraph with $|L| = |R| = m$. Suppose*

each vertex in $\mathcal{Y}$ is of degree n except for two vertices, $l \in L$ and $r \in R$, which are each of degree $n-1$. Then the induced subgraph on $L \cup R - \{l, r\}$, denoted by $\widehat{\mathcal{Y}}$, has a perfect matching.

Proof. Let A be any subset of $L - \{l\}$ of size $k < m$. We show that A has at least k distinct neighbors in $R - \{r\}$. By Hall's marriage theorem (see [88]), it will then follow that $\widehat{\mathcal{Y}}$ has a perfect matching. The total number of edges emanating from A regarded as a subset of the vertices of $\widehat{\mathcal{Y}}$ is at least

$$kn - (n-1) = (k-1)n + 1$$

since at most $n-1$ edges are incident to r in $\mathcal{Y}$. Hence, since the number of edges incident to a single vertex in $\widehat{\mathcal{Y}}$ is at most n, it follows that the number of distinct neighbors of A must be at least k. □ – Lemma 3

The algorithm operates by first finding the permutation to be routed on the faulty row. The existence of this permutation follows from Lemma 3. The edges of $\mathcal{B}$ involved in the corresponding matching are then deleted from $\mathcal{B}$. The resulting multigraph is regular since, by extracting these edges from $\mathcal{B}$, the degree of every vertex is reduced by 1, except for c_i and c'_i whose degrees remain the same. The algorithm now proceeds as described in the proof of Theorem 4.

Generalization to p faults. Suppose that a single copy of $\mathcal{G}$ in $\mathcal{G} \times \mathcal{H}$ contains $p > 1$ faults. We repeatedly apply a variant of Lemma 3 to find a permutation to route on each row that contains one of the faults in the faulty copy of $\mathcal{G}$.

Lemma 4 *Suppose n and p are integers such that $p \leq n$. Let $\mathcal{Y} = (L \cup R, E)$ be a bipartite multigraph with $|L| = |R| = m$. Suppose each vertex in $\mathcal{Y}$ is of degree $n - i$, where $0 \leq i \leq p-1$, except for two vertices, $l \in L$ and $r \in R$, each of degree $n - p$. Then the induced subgraph on $L \cup R - \{l, r\}$ has a perfect matching.*

Proof. Apply the same argument as that in Lemma 3. □ – Lemma 4

For each row that contains a fault, we can apply Lemma 4 to find some permutation that can be lifted to this row. We repeat this p times (once for each fault). After we have found the appropriate permutations to be routed on the faulty rows, the algorithm proceeds as before since, after the edges associated with the permutations for the faulty rows are removed, $\mathcal{B}$ is still a regular graph. □ – Theorem 10

Applying this result to specific networks is straightforward. For example, routing on an $n \times n$ mesh network (with wraparound) containing a single fault can be achieved in $3n-3$ steps, *i.e.*, without any time loss compared to the fault-free application of the direct product routing algorithm. For the n-dimensional hypercube network, we apply Theorem 10 inductively to a direct product of n copies of the complete graph on two vertices to achieve the desired result. Also, if one considers the hypercube routing implemented on the Beneš network, then this result shows that if any single column of switches fails then any permutation of the remaining live input nodes can still be achieved.

4.2 A General Theorem on the Connectivity of Cayley Graphs

In this section, we study the connectivity of Cayley graphs and derive as corollaries two previous results that have appeared in the literature. Due to the constructive nature of the main theorem, our result can be employed for studying the fault-tolerance properties of Cayley networks.

By definition, the **connectivity** of a graph, $\mathcal{G}$, denoted by $\kappa(\mathcal{G})$, is the minimum number of vertices that must be removed in order to disconnect the graph into disjoint (non-empty) subgraphs. The problem of determining the connectivity of highly symmetric graphs has been studied by several authors in recent years [56,66,49,51,75,106,107,108]. Much of the effort has gone into determining which vertex-transitive graphs have con-

nectivity equal to their degree. The most general result currently known was proved by Godsil [49]. He showed that every Cayley graph with a minimal generating set possesses optimal connectivity[1]. Akers and Krishnamurthy [3] showed that quasi-minimal Cayley graphs with a size restriction have optimal connectivity, and in particular, they proved that if the generating set is quasi-minimal and all of the generators are of order 2, then the resultant Cayley graph has optimal connectivity.

Our main result is the following: Let $H = \mathrm{gp}(S_H)$, $s \notin S_H$ and $G = \mathrm{gp}(S)$, where $S = S_H \cup \{s\}$. Then, if $|H| \geq 2(\kappa(\Gamma(H, S_H)) + \lambda_s - 1)$, we have

$$\kappa(\Gamma(G, S)) \geq \kappa(\Gamma(H, S_H)) + \lambda_s,$$

where λ_s is defined as

$$\lambda_s = \begin{cases} 1 & \text{if } o(s) = 2 \\ 2 & \text{if } o(s) > 2 \end{cases}$$

We derive both Godsil's result and Akers and Krishnamurthy's result as corollaries to this theorem. Our technique is constructive, based on the connectivity of induced Cayley graphs of subgroups and left coset graphs, and hinges on the structure theorem for left coset graphs proved in Chapter 2.

4.2.1 Preliminaries

We appeal to the following equivalent formulation of the notion of connectivity (see [31]):

> Γ has connectivity $\kappa(\Gamma)$ if and only if for any pair of vertices x, y and for any subset of vertices $A \subset V(\Gamma) - \{x, y\}$ of size $\kappa(\Gamma) - 1$, there exists a path in Γ between x and y that does not pass through any vertices of A. That is, there is a path between x and y that **avoids** A.

A is said to be an **avoidance set**. In the case that $\rho(\Gamma) = \kappa(\Gamma)$, Γ is said to have **optimal connectivity**.

[1]He proved this result for infinite as well as finite groups. Our results pertain only to finite groups.

The connectivity of vertex-transitive graphs has been studied in a general context by Green [51] and Watkins [106,107]. In particular, Watkins showed the following lower bound on the connectivity of an arbitrary vertex-transitive graph:

Theorem 11 (Theorem 3, [107]) *If Γ is a vertex-transitive graph then*

$$\kappa(\Gamma) \geq \frac{2}{3}(\rho(\Gamma) + 1).$$

For vertex-transitive graphs of small degree, Watkins [107] infers the following corollaries to this theorem.

Corollary 1 *Any vertex-transitive graph Γ having $\kappa(\Gamma) = 2$ is a polygon.*

This corollary implies that every degree 3 vertex-transitive graph has optimal connectivity.

Corollary 2 *If Γ is a vertex-transitive graph with $\rho(\Gamma) = 2$, 4 or 6, then Γ has optimal connectivity.*

As an aside, we mention that Watkins [108] has completely characterized the connectivity of Cayley graphs of cyclic groups (also known as circulants).

4.2.2 Proof of the Main Theorem

In this section we prove the following theorem.

Theorem 12 *Let S_H be a generating set for the subgroup H of G and suppose $S = S_H \cup \{s\}$, where $s \notin H$, generates G. Then, if $|H| \geq 2(\kappa(\Gamma(H, S_H)) + \lambda_s - 1)$, we have $\kappa(\Gamma(G, S)) \geq \kappa(\Gamma(H, S_H)) + \lambda_s$, where*

$$\lambda_s = \begin{cases} 1 & \text{if } o(s) = 2 \\ 2 & \text{if } o(s) > 2 \end{cases}$$

Proof. For convenience, we will use the following abbreviated notation for the duration of the proof.

- $\Gamma(G) = \Gamma(G, S)$
- $\Gamma(H) = \Gamma(H, S_H)$
- $\Gamma_L(G/H, S) = \Gamma_L(G/H)$
- $\kappa_H = \kappa(\Gamma(H, S_H))$

Recall the following two basic facts from Chapter 2.

- $\Gamma(G)$ consists of $|G|/|H|$ isomorphic copies of $\Gamma(H)$ connected by edges all of which are labeled by s.
- If s is of order 2 then $\rho(\Gamma(G)) = \rho(\Gamma(H)) + 1$, otherwise $\rho(\Gamma(G)) = \rho(\Gamma(H)) + 2$.

Consider the left coset graph, $\Gamma_L(G/H)$. Recall the structure theorem (Theorem 2) for left coset graphs presented in Chapter 2. Since $s \notin H$, it is evident that

$$d_i d_c \geq |H|. \tag{4.1}$$

Pick an arbitrary vertex x in $\Gamma(G)$ and suppose A is a subset of vertices of $\Gamma(G)$ such that $e, x \notin A$ and either $|A| = \kappa_H$ or $|A| = \kappa_H + 1$, depending on whether s is of order greater than 2 or equal to 2, respectively (we handle both cases simultaneously). The proof proceeds by constructing a path from e to x that avoids A. This is clearly sufficient since $\Gamma(G)$ is vertex-transitive. We accomplish this by using the connectivity of $\Gamma(H)$ together with the connectivity of the vertex-transitive graph $\Gamma_L(G/H)$ implied by the putative lower bound in Theorem 11. Intuitively, because of (4.1), either the internal degree or the external degree must be "large" and in either case this provides us with the freedom to find a path of the required type. We rely heavily on the fact that the

size of each bundle (*i.e.*, the number of edges of $\Gamma(G)$ between adjacent cosets) is a fixed constant. The proof splits into several cases.

CASE 0: $|G : H| = 2$.

If there are exactly 2 cosets, then the Cayley graph $\Gamma(G)$ consists of 2 copies of $\Gamma(H)$ connected by either one or two perfect matchings (depending on whether s is of order 2 or greater than 2, respectively).

SUBCASE (A). Suppose $x \in H$ and s is of order 2. If there are at most $\kappa_H - 1$ members of A in H then by the connectivity of $\Gamma(H)$ there is a path (in $\Gamma(H)$) between e and x avoiding A. If all of A lies in H, then we can find a path (in the second coset) between s and xs that avoids A and hence a path between e and x that avoids A.

SUBCASE (B). Suppose $x \in sH$ and s is of order 2. If there are κ_H members of A in $\Gamma(H)$, then we can cross over to the second coset and find a path between s and x entirely within this coset. Similarly, if there are κ_H members of A in xH then we find a path (entirely within $\Gamma(H)$) from e to xs (and hence from e to x). On the other hand, if the members of A are spread out between the 2 cosets, then all we must show is that we can cross from one coset to the other. This is clearly possible as long as $|H| > \kappa_H$.

SUBCASE (C). Now suppose $x \in H$ and $o(s) > 2$. As before, if there are at most $\kappa_H - 1$ members of A in H then by the connectivity of $\Gamma(H)$ there is a path (in $\Gamma(H)$) between e and x avoiding A. On the other hand, if there are at least κ_H members of A in $\Gamma(H)$, then either s or s^{-1} is not an element of A. (Note that they are both in the second left coset.) Suppose w.l.o.g. that $s \notin A$. Since we can find a path between s and either xs or xs^{-1} avoiding A (in the second coset), we can obtain a path between e and x that avoids A.

SUBCASE (D). Finally, suppose that $x \in sH$ and $o(s) > 2$. If either coset contains at least κ_H members of A, then we can proceed from either e or x into the adjacent coset

in which there is at most a single member of A. Thus, unless $|H| = 2$, we can find a path between e and x avoiding A. Now assume that there are at most $\kappa_H - 1$ members of A in either coset. We must determine when we can cross over from some vertex in H to some vertex in sH. Suppose there are x members of A in H and y members of A in sH. Then, we can cross over from the one coset to the other if

$$|H| - x > y.$$

Since $x + y = \kappa_H + 1$, we deduce that $|H| > \kappa_H + 1$.

CASE 1: The purpose of Case 1 is to reduce the problem to the case in which there are at most κ_H vertices of A in any given coset. This allows us "mobility" inside each coset. Thus, for Case 1, we may assume A lies in at most 2 distinct cosets. (Otherwise, we can immediately assert that there are at most κ_H vertices in each coset.) Let $A = A_1 \cup \{a\}$ where A_1 is contained in a single coset and a is a single vertex in an arbitrary coset.

SUBCASE (A): $\Gamma_L(G/H)$ is a polygon.

Using the fact that A resides in at most 2 cosets, we first show that there exists some coset, say yH, such that

1. There is a path between e and y avoiding A, and
2. $yH \cap A = \emptyset$.

If s has order 2 then $|A| = \kappa_H$. If A is a subset of H then clearly, $y = s$ satisfies the assertion. Similarly, if A does not intersect H then $y = e$ satisifies the assertion. If A is not a subset of H then there are at most $\kappa_H - 1$ members of A in each coset since $|A| = \kappa_H$, proving the assertion.

On the other hand, suppose that s is of order greater than 2. Let H contain κ_H members of A. Then, there is some neighbor, say v, of e (either s or s^{-1}) that is not an element of A and the coset vH contains at most 1 member of A. If there are no members

of A in vH then $y = v$ satisfies the assertion; otherwise, there is some vertex in vH (which we can reach using the connectivity of $\Gamma(H)$) adjacent to an element of a coset yH that does not intersect A. (This follows since there are at least $|H|/2$ vertices of vH that are adjacent to a coset $yH \neq H$.)

By a similar argument, there must exist at least one coset, say wH, such that

1. There is a path between x and w avoiding A, and
2. $wH \cap A = \emptyset$.

Note that yH may equal wH. Now, if either A resides completely in a single coset or both A' and a lie between yH and wH on the same side of the polygon then there is obviously a path between e and x avoiding A. If A' and a are on opposite sides of the polygon, then we find a path between yH and wH on the side containing a. The single vertex a is easily avoided since there are at least 2 distinct edges between adjacent cosets.

SUBCASE (B): $\Gamma_L(G/H)$ is not a polygon.

If $\Gamma_L(G/H)$ is not a polygon, by Corollary 1 we infer that $\kappa(\Gamma_L(G/H)) \geq 3$. As before, there exists some coset, say yH, such that $yH \cap A = \emptyset$ and there is a path between e and y avoiding A; similarly, there exists some coset, say zH, such that $zH \cap A = \emptyset$ and there is a path between e and y that avoids A. Since $\kappa(\Gamma_L(G/H)) \geq 3$, there exists a path in $\Gamma_L(G/H)$ between yH and zH avoiding the 2 cosets that contain A. This path can easily be converted to a path between y and z in $\Gamma(G)$ since each coset is a connected subgraph of $\Gamma(G)$ and there are no members of A in any of the cosets along this path. Hence, there is a path between e and x in $\Gamma(G)$ avoiding A.

CASE 2: We are now free to assume that no coset contains more than $\kappa_H - 1$ vertices of A and thus, for any 2 vertices (that are not members of A) in a fixed coset, there is a path between them that avoids A. Notice that this path does not leave the given coset.

If $x \in H$, we can find a path between e and x in $\Gamma(H)$ avoiding A by simply using the connectivity of H. Thus, suppose w.l.o.g. that $x \notin H$. We require the following definitions:

(i). Let P be a path in the left coset graph, $\Gamma_L(G/H)$. A **lifting** of P is defined to be any path in $\Gamma(G)$ that passes through the same set of left cosets as P in exactly the same order.

(ii). We say that a subset $A' \subset A$ **saturates** an edge (uH, vH) of $\Gamma_L(G/H)$ if for each edge, (uh, vh), of $\Gamma(G)$ in the bundle between uH and vH, we either have $uh \in A'$ or $vh \in A'$.

Any subset of A that saturates an edge of $\Gamma_L(G/H)$ will prohibit the lifting of a path in $\Gamma_L(G/H)$ that passes through that edge. Conversely, notice that any path in $\Gamma_L(G/H)$ that does not have any saturated edges may be lifted to a path in $\Gamma(G)$ that avoids A.

We find a path between e and x that avoids A by first finding a path from *some* vertex in H to *some* vertex in xH avoiding A. This is sufficient since a path of this type can easily be extended to a path of the required type using the connectivity of $\Gamma(H)$.

SUBCASE (A). Suppose that for all $a \in A$, $a \notin H$ and $a \notin xH$.

In order to saturate an edge of $\Gamma_L(G/H)$, we must have at least d_i edges with one of their endvertices in A. This can involve at most 2 left cosets. Any path that avoids *either* of these cosets *entirely* can then be lifted to a path in $\Gamma(G)$ (assuming there are no other saturated edges along the path in $\Gamma_L(G/H)$). Thus, the maximum number of cosets we will need to avoid is

$$\frac{\kappa_H + 1}{d_i}, \tag{4.2}$$

since $\kappa_H + 1$ is the maximum size of A and it takes at least d_i elements of A to saturate an edge of $\Gamma_L(G/H)$.

By Watkins' theorem (Theorem 11) we can avoid at least

$$\Delta \stackrel{\text{def}}{=} \frac{2}{3}d_e - \frac{1}{3} \tag{4.3}$$

vertices of $\Gamma_L(G/H)$. Thus, in order to be able to lift some path in $\Gamma_L(G/H)$ to $\Gamma(G)$ avoiding A, Δ must be at least as large as the quantity in expression 4.2. That is,

$$\frac{2}{3}d_e - \frac{1}{3} \geq \frac{\kappa_H + 1}{d_i}. \tag{4.4}$$

Rewriting this inequality, our condition becomes

$$\frac{2}{3}d_i d_e - \frac{d_i}{3} \geq \kappa_H + 1. \tag{4.5}$$

The left hand side of this last inequality achieves a minimum when $d_i = |H|$. However, in this case, the left coset graph is a polygon and the argument in Case 1 applies. Since d_i must divide $|H|$ (recall that it represents the size of a subgroup of H), we may assume that $d_i \leq |H|/2$ and hence, after simplification, our final condition becomes

$$|H| \geq 2(\kappa_H + 1), \tag{4.6}$$

using the fact that $d_i d_e \geq |H|$.

Remark. Note that in the case where $o(s) = 2$, the right hand side of inequality 4.6 is $2\kappa_H$.

SUBCASE (B): When some members of A are in either H or xH (*i.e.*, the initial or final cosets) then the paths lifted from $\Gamma_L(G/H)$ in the preceding case may have an endpoint at one of these vertices. Thus, our new aim is to find a path between H and xH that does not start or end at a member of A and avoids all of the members of A that are outside $H \cup xH$.

Firstly, we show that without loss of generality, we may assume that H and xH are not adjacent in $\Gamma_L(G/H)$. Suppose, to the contrary, that H and xH are adjacent in $\Gamma_L(G/H)$. Either we can find a path from e to x directly by using the d_i edges that

connect H and xH or all of these edges have (at least) one of their endvertices in A. In the latter case, we consider all of the s-edges of vertices in H that do not have either of their endvertices in A. Let C denote the set of these endvertices not in H. The size of this set is at least

$$|H| - (\kappa_H + 1),$$

and must comprise at least $\lceil |C|/d_i \rceil$ distinct cosets. If any member $c \in C$ is in a coset not adjacent to xH then we have reduced the problem to finding a path between c and x where c and x are not in adjacent cosets, as desired. On the other hand, suppose every element of C is in a coset adjacent to xH. In this case, there are at least $|C|$ edges in $\Gamma(G)$ that can be used to get to xH from vertices in C. At least one of these edges must have both of its endvertices outside of A since $|C| > 0$ (as long as $|H| > \kappa_H + 1$). Hence, we can reach xH from H via a single intermediate coset. Once inside xH we can find a path to x using the connectivity of $\Gamma(H)$.

Now, assuming e and x are not in adjacent cosets, we consider two cases according to the order of s.

1. Suppose s is of order 2. If s is of order 2 then we proceed to reduce the problem to Case 2, Subcase (a) as follows. Let B denote the subset of A that lies in $H \cup xH$. Replace A by $A' = (A - B) \cup C$ where C is the set of vertices $\{hs : h \in B\}$, *i.e.*, the set of all s-neighbors of vertices in B. (Since, by assumption, H and xH are not adjacent, A' does not intersect $H \cup xH$.) Since s is of order 2, it is sufficient to avoid A' in order to avoid A. Note that $|A'| = |A|$ and H does not contain any members of A' and thus the problem reduces to Case 2, Subcase (a).

2. Suppose s is of order greater than 2. Up until this point in the proof, we have not used the s^{-1} edges that lead out of the copies of $\Gamma(H)$ when $o(s) > 2$. The following lemma investigates the structure of $\Gamma(G)$ in this instance.

Proposition 8 *If s is of order greater than 2, then one of the following statements holds:*

1. $\rho(\Gamma_L(G/H)) = 2d_e$.
2. *The number of parallel edges between 2 left cosets of H in G is either 0 or $2d_i$.*
3. *Each bundle is composed of 2 perfect matchings.*

Proof. Focus on the coset H (by transitivity, it suffices to consider a single coset). Recall the definition of the subgroup K of H (Theorem 2) and let $R = \{r_i\}$ be a left transversal for K in H.

Firstly, assume that Ks^{-1} is not contained in any of the cosets r_isH, for all i (*i.e.*, the s^{-1} edges leaving K lead to a completely new left coset of H in G). We claim that *every* set of the form r_iKs^{-1} leads to a new left coset of H in G, thus showing that Statement 1 of the proposition holds. To prove the claim, observe that if $r_iKs^{-1} = r_jKs$ for some i and j, then $Ks^{-1} \subseteq r_mKs$ for some m, contradicting the original assumption.

Now assume that Ks^{-1} is contained in some coset $rksH$, for some $k \in K$ and $r \in R$. Then, clearly, all left translates of Ks^{-1} (*i.e.*, the sets of the form $r'Ks^{-1}$, for $r' \in R$) are also contained in some coset $r''k'sH$ ($r'' \in R$, $k' \in K$). Notice that 2 distinct left translates of Ks^{-1} cannot be contained in the same left coset of H in G by definition of K.

There are two possibilities: Suppose $r = e$, *i.e.*, both the s^{-1} and the s-edges leading out of K lead to the same left coset of H in G. In this case, either there are 2 perfect matchings between K and Ks, namely, one perfect matching labeled by s and another labeled by s^{-1}, or Ks^{-1} does not intersect Ks. The latter situation implies that the number of edges between 2 left cosets of H in G is either 0 or $2d_i$ (observe that if $s^{-1} \in sH$, then $s^2 \in H$). Finally, if $r \neq e$, then this again implies that the number of

edges between 2 left cosets of H in G is either 0 or $2d_i$. □

Returning to the proof of the theorem, we consider each of the situations in Proposition 8 in turn.

1. $\rho(\Gamma_L(G/H)) = 2d_c$**:** If there is a saturated edge of $\Gamma_L(G/H)$ that has H (the situation for xH is analogous) as one of its endpoints, then two possibilities arise. Consider the bundle of edges that comprise the saturated edge. If the d_i members of A that are causing the edge to be saturated are distributed between the 2 sides of the bundle, then we avoid the coset adjacent to $\Gamma(H)$. If, on the other hand, all of the d_i members of A are in $\Gamma(H)$, then it becomes necessary to avoid the 2 left cosets (one formed by the s-edges and the other formed by the s^{-1}-edges) that are adjacent to H. In effect, any lifting of a path in $\Gamma_L(G/H)$ that does not pass through either of these 2 left cosets is sufficient to avoid the members of A in this bundle. It follows that the maximum number of cosets we must avoid is $2(\kappa_H + 1)/d_i$. However, because $\rho(\Gamma_L(G/H)) = 2d_c$ in this case, we can avoid (by Watkins' theorem again) at least

$$\frac{2}{3}(2d_c) - \frac{1}{3}$$

cosets. After simplifying, we conclude that we require

$$|H| \geq 2(\kappa_H + 1).$$

2. The number of parallel edges between two left cosets of H in G is either 0 or $2d_i$: In this case, since the internal degree has doubled, twice as many elements of A are required in order to saturate an edge of $\Gamma_L(G/H)$. Using the same argument as in the previous case, we conclude that we require

$$\frac{2}{3}d_c - \frac{1}{3} \geq \frac{\kappa_H + 1}{d_i},$$

and after simplification, the condition becomes

$$|H| \geq 2(\kappa_H + 1).$$

3. Each bundle is composed of 2 perfect matchings: In this case, the endpoints of the s-edges and the s^{-1}-edges that are not in H are in the *same* left coset. Thus, this case is equivalent to the case in which $o(s) = 2$. This completes the proof of the Theorem. □

From Theorem 12, we derive the following 2 corollaries.

Corollary 3 [3] *Any quasi-minimal Cayley graph where every generator is of order 2 has connectivity equal to the graph's degree.*

Proof. The proof is by induction on the number of generators. Suppose that the generating set S_H (in Theorem 12) is quasi-minimal and $|S_H| = k$. Suppose further, that by the inductive hypothesis $\Gamma(H, S_H)$ has connectivity equal to its degree, *i.e.*, $\kappa(\Gamma(H, S_H)) = k$. Then, by the result of Theorem 12, $\Gamma(G, S)$ has connectivity $k + 1$, since

$$|H| \geq 2^k \geq 2(\kappa(\Gamma(H, S_H))) = 2\kappa(\Gamma(H, S_H)),$$

for all $k \geq 2$. For the case $k = 1$, the connectivity of $\Gamma(G, S)$ is optimal by Corollary 2. □

Corollary 4 [49] *All minimal Cayley graphs have connectivity equal to their degree.*

Proof. Let $S_H = \{s_1, s_2, \ldots, s_k\}$ be a minimal generating set for $H = \text{gp}(S_H)$. Define $G = \text{gp}(S)$, where $S = S_H \cup \{s\}$ and S is a minimal. As in the previous corollary, the proof is by induction on the number of generators in S. We distinguish several cases.

CASE 1. Suppose that $o(s_1) = 2$. It follows that either $o(s_2) = 2$, or $o(s_2) > 2$ and $|\text{gp}(s_1, s_2)| > 4$, since S is minimal.

SUBCASE (A). If $o(s_1) = o(s_2) = 2$, then to ensure optimal connectivity of $\Gamma(G)$, we require (by Theorem 12) that

$$2^k \geq 4k - 2,$$

which is satisfied as long as $k \geq 4$. When $k = 1$, we apply Corollary 2. When $k = 3$ then either $\rho(\Gamma(H)) = 4$ or $\rho(\Gamma(H)) = 3$. We handle each of these cases in turn.

$\underline{\rho(\Gamma(H)) = 4.}$ If $\rho(\Gamma(H)) = 4$ and $o(s) = 2$ then we require (by Theorem 12) that

$$2^3 \geq 4(3),$$

which is obviously true. If $\rho(\Gamma(H)) = 4$ and $o(s) > 2$ then $\rho(\Gamma(G)) = 6$, and by Corollary 2, $\Gamma(G)$ has connectivity equal to its degree.

$\underline{\rho(\Gamma(H)) = 3.}$ If $\rho(\Gamma(H)) = 3$ and $o(s) = 2$, then $\rho(\Gamma(G)) = 4$ and thus $\Gamma(G)$ has connectivity equal to its degree by Corollary 2. If $\rho(\Gamma(H)) = 3$ and $o(s) > 2$, then we require (by Theorem 12) that

$$2^3 \geq 2(3) + 2,$$

which is obviously true.

SUBCASE (B). Suppose that $o(s_2) > 2$ and $|\mathrm{gp}(s_1, s_2)| > 4$. In this case, we require that

$$3(2^{k-1}) \geq 4k + 2,$$

which is satisfied if $k \geq 4$. If $k = 3$ and $o(s) = 2$, then we require that

$$3(2^2) \geq 4(3),$$

which is obviously true. On the other hand, if $k = 3$ and $o(s) > 2$, then $\rho(\Gamma(G))$ is either 6 or 7. If its degree is 6, then the connectivity of $\Gamma(G)$ is equal to its degree by Corollary 2. If its degree is 7, then the minimal size of H is 12. To prove our result in this situation, we require a fine analysis of the possibilities for the values of d_i and $\kappa(\Gamma_L(G/H))$. This analysis is summarized in the table below. For each row in the table, it is easy to check that the equation

$$\kappa(\Gamma_L(G/H)) \geq \frac{2(3)+1}{d_i}$$

holds, thus proving the assertion.

d_i	$\rho(\Gamma_L(G/H))$	$\kappa(\Gamma_L(G/H))$
4	3	3
3	4	4
2	6	6

Table 4.1: Values for important parameters when $|H| = 12$.

CASE 2. Suppose that $o(s_1) > 2$. Then, we require that

$$3(2^{k-1}) \geq 2(2k) + 2 = 4k + 2,$$

which is satisfied as long as $k \geq 4$. If $k = 3$ and $o(s) = 2$, then we require

$$3(2^2) \geq 4(3),$$

which is obviously true. The case when $o(s) > 2$ follows again from the above table used in the proof of Case 1. This concludes the proof that all minimal Cayley graphs have optimal connectivity. □

Remark on the Complexity of Avoiding Faults. Define an **avoidance algorithm** for a graph $\mathcal{G}$ to be an algorithm that, on presentation of a pair of vertices u, v and an avoidance set A of cardinality $\kappa(\mathcal{G}) - 1$, produces a path in $\mathcal{G}$ avoiding A. If we are given avoidance algorithms for the Cayley graph $\Gamma(H)$ and the left coset graph $\Gamma_L(G/H)$ then it is possible to construct an avoidance algorithm for $\Gamma(G)$. We simply "weave" a path through $\Gamma(G)$ according to the distribution of A (using the various cases of the proof) and call the avoidance algorithms for $\Gamma(H)$ and $\Gamma_L(G/H)$ as needed. The worst case time complexity is then easily calculated as

$$t_{G/H} + \max_{(u,v)\in E(\Gamma_L(G/H))} |P(u,v)| t_H$$

where $t_{G/H}$ and t_H are the time complexities of the avoidance algorithms for $\Gamma_L(G/H)$ and $\Gamma(H)$, respectively, and $P(u,v)$ is a path between u and v in $\Gamma_L(G/H)$ that avoids an arbitrary set of vertices[2].

[2] $E(\mathcal{G})$ denotes the edge set of the graph $\mathcal{G}$.

4.2.3 Applications to Fault-Tolerant Routing

A network's resilience to faults is often measured by the redundancy in the number of possible disjoint communication paths between pairs of distinct PEs. This is equivalent to determining the value of the underlying graph's connectivity: If the network has connectivity κ then the failure of any $\kappa - 1$ PEs does not preclude any pair of PEs from communicating. This is a common *worst case* measure of the fault-tolerance of a network [3,65,99], the limitations of which cannot be overcome except by improving or adding hardware. For example, a family of bounded degree networks can tolerate only a constant number of faults *independent* of the size of the network. However, the following fault-model presupposes a less stringent distribution:

> **Clustering of faults.** Due to the fabrication process, in VLSI systems it is common to find large numbers of failures occuring in localized groups. Hence, even though localized regions may be inoperable it is still desirable to maintain communication between "live" regions.

For quasi-minimal Cayley networks, the next proposition can be applied in this scenario.

Proposition 9 *Let $\Gamma(G,S)$ be a quasi-minimal Cayley graph as defined in the proof of Theorem 12. Let x and y represent 2 PEs in a network that has $\Gamma(G,S)$ as its underlying topology and suppose $\mathcal{F}$ is a set of faulty PEs that do not lie in the left cosets containing x and y. Then the following statements are true.*

1. *As long as $\mathcal{F}$ consists of at most $\kappa(\Gamma_L(G/H))$ distinct cosets, there is a path between x and y avoiding all of the cosets containing elements of $\mathcal{F}$.*

2. *If $|\mathcal{F}| \leq \kappa(\Gamma_L(G/H)) + d_i - 2$, there is a path in the network between x and y avoiding $\mathcal{F}$.*

Proof. The proof of the first assertion is straightforward since any path in $\Gamma_L(G/H)$ avoiding a specific set of cosets can be lifted to a path in $\Gamma(G)$ that avoids the same set of cosets.

For the proof of the second assertion, observe that we can avoid an arbitrary $\kappa(\Gamma_L(G/H)) - 1$ faulty PEs in $\Gamma(G)$ due to the connectivity of $\Gamma_L(G/H)$ and another $d_i - 1$ PEs along any lifted path of $\Gamma_L(G/H)$. The result follows. □

Example 1: Wreathed Networks

In the first example, we illustrate our approach to connectivity by considering $\mathcal{W}_{n,k}$, the order-(n,k) wreathed network, defined in Chapter 2. By Theorem 3, we have that the internal degree of $\mathcal{W}_{n,k}$ is n/k and the external degree is k. Also, the left coset graph, $\Gamma_L(G_{n,k}/H_{n,k}, \{\tau_n, \sigma_{n,k}\})$, was found to be isomorphic to the k-dimensional hypercube and by Corollary 3, the connectivity of the k-dimensional hypercube is equal to k. Thus, by Proposition 9, we can avoid any $n/k + k - 2$ vertices that lie between 2 distinct left cosets of $H_{n,k}$ in $G_{n,k}$.

Example 2: Generalized Ring Networks

Representing the opposite extreme in comparison to the cube-connected cycles network, consider an arbitrary quasi-minimal Cayley graph of an *abelian* group, denoted by G. This class of graphs generalizes the interconnection pattern of mesh-like networks (such as the toroidal mesh, the hypercube and many double ring networks). Let H be the subgroup of G generated by all of the generators except for the last one, which we denote by s. Assume, for the sake of simplicity, that the order of s is greater than 2. The graph $\Gamma_L(G/H)$ is easily seen to be an ℓ-vertex ring where ℓ is the order of s. Thus, the external degree in this case is 2 and the internal degree is $|H|$. Hence, appealing to Proposition 9, we obtain a method for routing around faults in such networks by recursively reducing

the problem to routing in ring networks.

4.2.4 Quotient Connectivity

The aim of the final topic of this section is to introduce an extension to the notion of connectivity for the purpose of studying a network's resilience to faults under an alternative fault model in which faults are assumed to occur in clusters. Instead of using the connectivity of the network itself we use the connectivity of an appropriate "quotient" graph. In its most general form, our idea can be expressed as follows. Let $\mathcal{G}$ be a graph and suppose

$$\mathbf{P} = V_1 \cup V_2 \cup \cdots \cup V_k$$

is a partition of the vertex set of $\mathcal{G}$ for which the subgraph induced by each V_i is connected. Form a new graph, denoted by $\mathcal{G}/\mathbf{P}$, whose vertices are the blocks of $\mathbf{P}$ (*i.e.*, each set V_i is a vertex of $\mathcal{G}/\mathbf{P}$) and there is an edge between 2 blocks V_i and V_j when some pair of vertices (v_i, v_j), for $v_i \in V_i$ and $v_j \in V_j$, is an edge of $\mathcal{G}$. Notice that when $\mathcal{G}$ is a Cayley graph of a group G (with generating set S) and the partition $\mathbf{P}$ consists of the set of left cosets with respect to some subgroup H of G, then we have $\mathcal{G}/\mathbf{P} = \Gamma_L(G/H, S)$.

One possible approach is to partition the network into *balls*[3] of (more or less) equal size and examine the connectivity of the resulting quotient. Although this appears to be an interesting combinatorial problem in general and may yield interesting results for graphs such as expanders, we do not pursue it further here.

A second approach corresponds to the hierarchy of components often found in the construction of a parallel network. As an example we consider the 64K-PE connection machine [60]. At the lowest level is a VLSI *chip* consisting of 16 *cells* (the processing elements); 32 of these are connected together to form a *module*, 16 modules go together to make up a *backplane* and two modules comprise a single rack. The topology of each

[3]The *ball* of radius r around a vertex v is the set of all vertices within distance r of v.

level in this hierarchy is a hypercube. Our model of fault-tolerance allows the failure of sets of units at any level of the hierarchy. For example, given the failure of any 11 chips, the remaining non-faulty chips may still communicate with each other. This idea can be put in an entirely general context by recalling the structure of quasi-minimal Cayley networks. They are constructed by recursively connecting *subunits*, each of which is a Cayley network of a subgroup of the original group (in accord with the structure theorem for left coset graphs). In this scenario, we may apply Proposition 9, which provides us with a measure of how many subunits can fail at each level until communication capabilities between live subunits is severed. Note that these results suggest a fault-tolerant approach for the packaging of quasi-minimal Cayley networks, in the same fashion as that of the Connection machine.

CHAPTER 5

EMBEDDING CAYLEY GRAPHS INTO DIRECT PRODUCT GRAPHS

In this chapter, we show that every Cayley graph, $\Gamma(G, S)$, of a non-prime order group, G, can be embedded in a direct product of two smaller graphs with dilation 2 and congestion $O(|S|)$. When G has an appropriate subgroup structure then one has the choice of a number of direct product graphs, some of which preserve (to within a factor of 2) the degree of $\Gamma(G, S)$. The factors of the direct product graphs are either Cayley graphs or coset graphs and in certain cases, we can efficiently determine, via a linear-time algorithm, a degree-preserving embedding. We illustrate these embeddings by considering the butterfly-derivative networks. In the latter part of this chapter, we use the embeddings to find *work-preserving* emulations of Cayley networks on a rich family of smaller "quotient" networks, thereby exposing a processor-time tradeoff for Cayley networks.

5.1 The Embedding Results

We present two methods for embedding a Cayley graph in a direct product. The first is completely general while the second only applies to groups with a specific subgroup structure. Define a **2-embedding** to be a dilation 2, *surjective* graph embedding[1].

[1]Hence, if the domain and range of the embedding are equinumerous, then a 2-embedding is bijective.

5.1.1 The Schreier Technique

Suppose S generates G and H is any non-trivial subgroup of G. The following proposition constructs a generating set for H in terms of S and a right transversal for H in G.

Proposition 10 (O. Schreier [89]) *Let T be a right transversal for H in G. Then, the set*

$$\Delta(H; T, S) = \{ ts\overline{ts}^{-1} : t \in T, s \in S \}$$

is a generating set for H.

The elements of $\Delta(H; T, S)$ are called the **Schreier generators for H with respect to T.**

Theorem 13 *Suppose $G = \mathrm{gp}(S)$ and H is a non-trivial subgroup of G. Choose a right transversal T for H in G. Then, $\Gamma(G, S)$ can be 2-embedded in $\Gamma(H, \Delta(H; T, S)) \times \Gamma_R(G/H, S)$ with congestion $O(|S|)$.*

Proof. Let $S = \{s_1, s_2, \cdots, s_n\}$ and suppose $|T| = m$. Each element of G can be expressed as the product of an element of H and an element of T (since H is a subgroup and T is a right transversal for H in G). Thus, for each $g \in G$, let $g = h_g \overline{g}$ where $h_g \in H$ and $\overline{g} \in T$. As we have seen (in Chapter 2), this decomposition is unique.

Define an embedding (ϕ, ρ) of $\Gamma(G, S)$ in $\Gamma(H, \Delta(H; T, S)) \times \Gamma_R(G/H, S)$ as follows. The assignment map $\phi : G \to H \times G/H$ is specified by

$$\phi(g) = (h_g, \overline{g}).$$

ϕ is a bijection since the decomposition is unique. Now consider the effect that right multiplication by elements of S has on elements of T. Let $t \in T$ and $s_i \in S$. Since $ts_i \in H\overline{ts_i}$, there exists an element $h(t, i) \in H$ such that

$$ts_i = h(t, i)\overline{ts_i}.$$

Note that the set $\{h(t,i) : t \in T,\ i = 1, 2, \cdots, n\}$ is the set of Schreier generators for H with respect to T.

In order to define the routing function ρ, consider a specific edge (g, gs_i) of $\Gamma(G,S)$. If $g = h_g\overline{g}$, then

$$\begin{aligned} gs_i &= h_g\overline{g}s_i \\ &= h_g h(\overline{g}, i)\overline{\overline{g}s_i} \end{aligned}$$

Note that $\overline{\overline{g}s_i}$ resides in the same coset modulo H as $\overline{g}s_i$. Thus, we define the image of an edge (g, gs_i) to be the edge $(h_g, h_g h(\overline{g}, i))$ in $\Gamma(H, \Delta(H; T, S))$ followed by the edge $(\overline{g}, \overline{g}s_i)$ in $\Gamma_R(G/H, S)$, *i.e.*, one edge in the first factor of the direct product followed by one edge in the second factor of the direct product. For convenience, we refer to an edge in the first factor (*i.e.*, in $\Gamma(H, \Delta(H; T, S))$) as an $\mathcal{F}_1$ edge and similarly, we refer to an edge in the second factor (*i.e*, in $\Gamma_R(G/H, S)$) as an $\mathcal{F}_2$ edge. Note that because of the commutativity of the direct product operation, we could reverse the order of the edges in the factors without changing the dilation of the embedding. We can use this freedom in order to minimize the congestion. (In the worst case, this may only reduce the congestion by a constant factor.)

Suppose that we first route $\mathcal{F}_1$ edges followed by $\mathcal{F}_2$ edges. Let (g, g') be an edge of $\Gamma(G,S)$ and suppose the image of (g, g') under ρ is the edge f_1 of $\mathcal{F}_1$ followed by the edge f_2 of $\mathcal{F}_2$. The congestion on f_2 can be at most 2 since the embedding is bijective. (This follows because an edge (u, v) of the direct product may be used to get from u to v *or* from v to u.) The congestion that can occur on f_1 depends solely on the number of edges of $\Gamma(G,S)$ originating at either g or g' that "collapse" to f_1 in the image of ρ. Clearly, this is at most twice the degree of $\Gamma(G,S)$. (See the remark below.) Since the degree of $\Gamma(G,S)$ is $O(|S|)$, the theorem follows. □

A Remark on Congestion. We obtain a precise measure of the congestion occurring along a single edge of $\Gamma_R(G/H, S)$ as follows: Two edges (g, gs_i) and (g, gs_j) of $\Gamma(G, S)$ $(s_i, s_j \in S)$ collapse to the same edge of $\Gamma_R(G/H, S)$ if and only if $gs_i = hgs_j$ for some $h \in H$. This implies that s_i and s_j are in the same coset modulo $g^{-1}Hg$. Conversely, if 2 generators s_i and s_j are in the same coset modulo $g^{-1}Hg$, then the 2 edges (g, gs_i) and (g, gs_j) of $\Gamma(G, S)$ collapse to the same edge of $\Gamma_R(G/H, S)$. It follows that the congestion in $\Gamma_R(G/H, S)$ cannot exceed the maximum number of $s \in S$ that reside in a single right coset of some $g^{-1}Hg$ as g ranges over G.

5.1.2 Orthogonal Subgroup Technique

In the Schreier technique, the degree of one of the factors may be as large as $|S||G|/|H|$, which is often substantially larger than the degree of the original Cayley graph. However, when G has two subgroups of the appropriate sizes, we can employ a different method to obtain an embedding in which the degree of each factor is no larger than the degree of the original graph.

Theorem 14 *Suppose H and K are subgroups of G such that $|G|/|K| = |H|$. Then, $\Gamma(G, S)$ can be 2-embedded in $\Gamma_R(G/K, S) \times \Gamma_R(G/H, S)$ with load factor $|K \cap H|$ and congestion $O(|K \cap H||S|)$.*

Proof. Define an embedding (ϕ, ρ) as follows. The assignment map ϕ :G$\rightarrow$ $G/K \times G/H$ is given by

$$\phi(g) = (Kg, Hg).$$

Firstly, we show that ϕ is a $|K \cap H|$-to-one map. Suppose that $\phi(g_1) = \phi(g_2)$ for some $g_1, g_2 \in G$. Then $Kg_1 = Kg_2$ and $Hg_1 = Hg_2$ and hence $g_1 g_2^{-1} \in K \cap H$. This establishes the claimed load factor.

To define the routing function ρ, suppose (g, gs) is an edge of $\Gamma(G, S)$. Then, if $\phi(g) = (Kg, Hg)$, it follows that

$$\phi(gs) = (Kgs, Hgs).$$

Thus, we define the image of the edge (g, gs) in $\Gamma(G, S)$ under ρ to be the pair of edges

$$[(Hg, Kg), (Hgs, Kg)] \text{ and } [(Hgs, Kg), (Hgs, Kgs)]$$

in $\Gamma_R(G/N, S) \times \Gamma_R(G/H, S)$. The stated congestion bound follows from the same argument as in the Schreier embedding technique. □

We may apply this embedding technique in a wide variety of circumstances, as the following corollary indicates.

Corollary 5 *Suppose G is the semidirect product of N by H, generated by S. Then, there is a 2-embedding of $\Gamma(G, S)$ in $\Gamma(H, S') \times \Gamma_R(G/H, S)$, where S' denotes the projection of S into the factor group G/N.*

Proof. By definition, $H \cap N = \{e\}$ and $G/N \simeq H$. Hence, H and N are orthogonal subgroups. Therefore, the result follows by Theorem 14. □

5.1.3 Independent Generator Technique

In this section we study a particular situation wherein we can test in linear time whether a Cayley graph has a bounded degree embedding into a direct product graph. This embedding arises explicitly from the structure of the generating set rather than the subgroup structure of the group. We first require several definitions.

Suppose S generates the group G. Let S^* denote the free monoid over the symbols in S, *i.e.*, the set of all finite words over the symbols in S, where null represents the empty word. For $x, y \in S^*$, we write

- $x =_* y$ if x and y are equal *as words*, *i.e.*, as elements of S^*.

- $x =_{gp(S)} y$ if x and y are equal as elements of $gp(S)$.

For $A \subset S$, define the **projection map**

$$\pi_A : S^* \rightarrow S^*$$

via the inductive extension of $\pi_A : S \rightarrow A \cup \{\text{null}\}$:

$$\pi_A(s) = \begin{cases} s & \text{if } s \in A \\ \text{null} & \text{if } s \in S - A \end{cases}$$

$A \subset S$ is **G-independent**, where $G = gp(S)$, if the projection map π_A, regarded as a map from G into $gp(A)$, obeys the following **independence condition**: For all words $x, y \in S^*$,

$$x =_{gp(S)} y \Rightarrow \pi_A(x) =_{gp(S)} \pi_A(y).$$

We can detect whether A is G-independent if $\Gamma(G, S)$ is given by an adjacency list as the following proposition shows.

Proposition 11 *Suppose $\Gamma(G, S)$ is given by an edge-labeled adjacency list. Given any subset $A \subset S$, one can determine whether or not A is G-independent in $O(|G||S|)$ steps, where each step is either a multiplication or equality test in G.*

Proof. The following procedure performs a breadth-first search of $\Gamma(G, S)$ and attempts to label it appropriately in order to discover whether A is G-independent.

Algorithm Independence_Test()

- **Step 1**: Label e_G with e_G and all other vertices undefined.
- **Step 2**: Inductively construct a breadth-first tree of $\Gamma(G, S)$: For each new edge (u, v) with $us = v$ for some $s \in S$ do the following.
  - If $s \in A$ then $\text{temp}(v) = \text{label}(u)s$,

- Else if $s \in S - A$ then $\mathsf{temp}(v) = \mathsf{label}(u)$.
- If $\mathsf{label}(v) == \mathsf{undefined}$ then $\mathsf{label}(v) = \mathsf{temp}(v)$,
- Else if $(\mathsf{label}(v) \neq \mathsf{temp}(v))$ then **return** "Not Independent"

- **Step 3: return** "Independent"

Validation. The labels produced by the algorithm for the vertices of $\Gamma(G, S)$ are precisely the values of the projection map π_A, regarded as elements of $\mathrm{gp}(A)$.

If the algorithm returns "Not Independent", then it has tried to label some node with two distinct labels. Thus, in this case there are two distinct words $x, y \in S^*$ for which $x =_{\mathrm{gp}} y$ but $\pi_A(x) \neq_{\mathrm{gp}} \pi_A(y)$, *i.e.*, the independence condition is not satisfied.

Conversely, if the algorithm returns "Independent", then the labelling produced (and hence π_A) is well-defined. (Each vertex has received a single label.) Suppose, in this case, that $x, y \in S^*$ such that $x =_{\mathrm{gp}} y$. We must verify that $\pi_A(x) =_{\mathrm{gp}} \pi_A(y)$. Since $x =_{\mathrm{gp}} y$, the paths in $\Gamma(G, S)$ defined by x and y terminate at the same vertex in $\Gamma(G, S)$. Also, since the algorithm returned the result "Independent", the labels corresponding to x and y are identical. Since these labels are, respectively, $\pi_A(x)$ and $\pi_A(y)$, considered as elements of $\mathrm{gp}(A)$, we have that $\pi_A(x) =_{\mathrm{gp}} \pi_A(y)$.

Timing Analysis. Assuming that forming products and testing for equality of labels takes time $O(1)$, the time for the algorithm is dominated by the breadth-first traversal of $\Gamma(S)$, which takes time linear in the number of edges of $\Gamma(G, S)$, *i.e.*, $O(|G||S|)$. □

Our next aim is to present a purely group-theoretic characterization of the structure of a group satisfying the independence condition. For $B \subset G$, the group generated by the set of all conjugates[2] of elements in B by elements in G is called the **normal closure** of B in G and is denoted by $\mathrm{Ncl}(B)$. $\mathrm{Ncl}(B)$ is the smallest normal subgroup of G containing B.

[2]For two group elements $h, g \in G$, the conjugate of h by g is the product $g^{-1}hg$.

Lemma 5 *A set $A \subset S$ is G-independent if and only if G is the semidirect product of* $\mathrm{Ncl}(\mathrm{gp}(S-A))$ *by* $\mathrm{gp}(A)$.

Proof. Set $B = S - A$ and suppose g is an arbitrary element of G expressed as a product of elements from S. We can rewrite g in the form $a \cdot n$ where $a \in \mathrm{gp}(A)$ and $n \in \mathrm{Ncl}(B)$ by repeatedly "pushing" generators from A to the left (starting from the right) using the following generic equation:

$$ba = aa^{-1}ba$$

$(a \in A, b \in B)$. Note that $a^{-1}ba \in \mathrm{Ncl}(B)$. It remains to be shown that $g \in G$ can be expressed *uniquely* in the form $a \cdot n$. To this end, suppose that $g = a \cdot n = a' \cdot n'$. Then, since A is independent in G, we have

$$\begin{array}{lll} \pi_A(a \cdot n) & =_{\mathrm{gp}(S)} \pi_A(a' \cdot n') & \\ \Rightarrow a & =_{\mathrm{gp}(S)} a' & \text{(using the independence condition)} \\ \Rightarrow n & =_{\mathrm{gp}(S)} n' & \end{array}$$

Hence an arbitrary element of G can be uniquely written in the form $a \cdot n$ and thus, by definition, G is the semidirect product of $\mathrm{Ncl}(B)$ by $\mathrm{gp}(A)$. □

We now combine the results of this section with the orthogonal embedding technique to obtain the following theorem.

Theorem 15 *Let S generate G and suppose that $A \subset S$ is G-independent. Then $\Gamma(G,S)$ can be 2-embedded in $\Gamma(\mathrm{gp}(A), A) \times \Gamma_R(G/\mathrm{gp}(A), S)$ with congestion $O(|S|)$*

Proof. By Lemma 5, if A is G-independent then G is a semidirect product of $\mathrm{Ncl}(B)$ by $\mathrm{gp}(A)$, where $B = S - A$. Hence, the theorem follows by Corollary 5. □

5.1.4 An Illustration – Butterfly-Like Networks

We illustrate the embedding techniques by considering the wreath product, $Z_2 \wr Z_n$. Recall the following facts from Chapter 2:

- $\mathcal{B}_n = \Gamma(Z_2 \wr Z_n, \mathbf{C})$ and $\mathcal{D}_n = \Gamma_R(Z_2 \wr Z_n/Z_n, \mathbf{C})$
- $\mathcal{C}_n = \Gamma(Z_2 \wr Z_n, \mathbf{S})$ and $\mathcal{S}_n = \Gamma_R(Z_2 \wr Z_n/Z_n, \mathbf{S})$

For definiteness, we define

$$\begin{aligned} \sigma_1 &= \langle 1; 0, 0, \ldots, 0 \rangle \\ \sigma_2 &= \langle 1; 0, 0, \ldots, 1 \rangle \\ \sigma_3 &= \langle 0; 0, 0, \ldots, 1 \rangle, \end{aligned}$$

and note that $\mathbf{C} = \{\sigma_1, \sigma_2\}$ and $\mathbf{S} = \{\sigma_1, \sigma_3\}$ are the generating sets for the butterfly graph and the cube-connected cycles graph, respectively.

The Schreier Technique

Consider the subgroup $N = Z_2^n$ of $G = Z_2 \wr Z_n$. This subgroup is the set of all elements of $Z_2 \wr Z_n$ with a 0 in the first component. Define the following right transversal of N in G:

$$T = \{\langle \alpha; 0, 0, \cdots, 0 \rangle : \alpha \in Z_n\}.$$

The Schreier generators, $\Delta(N; T, \mathbf{C})$, are all of the elements of the form

$$t\sigma_i \overline{t\sigma_i}^{-1},$$

for $i = 1, 2$. If $i = 1$, then for each $t \in T$, $t\sigma_1 \overline{t\sigma_1}^{-1} = \langle 0; 0, 0, \cdots, 0 \rangle$, *i.e.*, the identity element of G. If $i = 2$, then $t\sigma_2 \overline{t\sigma_2}^{-1}$ is the set of all elements with a 0 in the first component and a single 1 in the second component. Hence, the Cayley graph $\Gamma(N, \Delta(N; T, \mathbf{C}))$ is easily seen to be isomorphic to the n-dimensional hypercube.

A similar argument discovers that $\Delta(N; T, \mathbf{S}) = \Delta(N; T, \mathbf{C})$ and hence $\Gamma(N, \Delta(N; T, \mathbf{S}))$ is also the n-dimensional hypercube.

The graph $\mathcal{R}_n = \Gamma_R(G/N, \{\sigma\})$ can easily be seen to be isomorphic to an n-vertex cycle. Hence, by Theorem 13, we obtain 2-embeddings of both $\mathcal{B}_n$ and $\mathcal{C}_n$ in $\mathcal{H}_n \times \mathcal{R}_n$.

The Orthogonal Subgroup Technique

Recall that the wreath product $Z_2 \wr Z_n$ is a semidirect product of Z_2^n by Z_n; hence, we have that Z_2^n and Z_n are orthogonal subgroups. Therefore, by applying Theorem 14, we obtain a 2-embedding of $\mathcal{B}_n$ in

$$\Gamma_R(Z_2 \wr Z_n/Z_2^n, \mathrm{C}) \times \Gamma_R(Z_2 \wr Z_n/Z_n, \mathrm{C}),$$

which is easily seen to be isomorphic to $\mathcal{R}_n \times \mathcal{D}_n$. This embedding forms the first step in an $O(\log n)$ embedding of the order-n butterfly graph in an order-$(n + \log n)$ deBruijn graph presented in [10]. By a similar application of Theorem 14, we obtain a 2-embedding of $\mathcal{C}_n$ in $\mathcal{R}_n \times \mathcal{S}_n$.

Independent Generator Technique

Consider the subset $A = \{\sigma_1\}$ of S. It is $Z_2 \wr Z_n$-independent since σ_1 is the only generator that affects the first component of any word in $Z_2 \wr Z_n$. Hence, by Theorem 15, $\mathcal{C}_n$ can be 2-embedded in

$$\Gamma(\mathrm{gp}(A), A) \times \Gamma_R(Z_2 \wr Z_n/\mathrm{gp}(A), \mathrm{S}).$$

The first factor is obviously isomorphic to $\mathcal{R}_n$ while the second is isomorphic to $\mathcal{S}_n$. (Hence, the independent generator technique and the complementary subgroup technique yield the same embeddings.)

On the other hand, A is *not* group-independent in C, as the following argument shows. Let

$$x_n = {\sigma_2}^{n-1}\sigma_1{\sigma_2}^{n-1}, y_n = {\sigma_1}^{n-1},$$

and observe that $x_n =_{gp} y_n$, even though

$$\pi_A(x_n) =_{gp} \sigma_1 \neq_{gp} {\sigma_1}^{n-1} =_{gp} \pi_A(y_n), \text{ when } n > 2.$$

5.2 Emulations of Cayley Networks

A common approach to comparing the power of competing topologies is to determine how efficiently one can *emulate* another on general computations [26,69,97]. An important theme in this study is to find small networks that can "efficiently" emulate larger ones, thereby trading off performance for hardware [22,46]. In particular, we would like to find emulations that are *work-preserving*: A guest network $\mathcal{G}$ is said to have a **work-preserving emulation** on a host network $\mathcal{H}$ if any T computation steps of $\mathcal{G}$ can be emulated in $O(T|G|/|H|)$ steps on $\mathcal{H}$, *i.e.*, the *processor-time* product is preserved (up to contant factors). Whereas previous work along these lines has centered on ad hoc analyses of the structure of specific networks [46] or on graph grammers [22], our approach builds on the group-theoretic specification of networks.

A second motivation for this problem relating specifically to Cayley networks arises from the unfortunate fact that many commonly proposed Cayley networks are so enormous that their practicality is called into question. For example, the order-n star network (proposed by Akers and Krishnamurthy [4]) has order $n!$, making it practically infeasible for large n. Nevertheless, these networks are attractive topologies due to their small diameter, high connectivity and amenability to efficient algorithm design. Hence, it is of practical interest to construct scaled-down versions of these networks while ensuring that they retain the ability to efficiently execute algorithms designed for the original network. The emulation technique presented here provides a positive response to this problem, and supplies further evidence in support of the use of Cayley networks for algorithm design due to the generality of the approach and the efficiency of the emulations produced.

For general graphs, the problem of finding work-preserving emulations has been shown

to be NP-complete [29,30]. In fact, in many cases, even if the emulating graph is *fixed*, the problem remains NP-complete. Our results show that, given a Cayley network of non-prime order, $\mathcal{G}$, there exists a family of smaller networks that can efficiently emulate $\mathcal{G}$ on general computations. We now describe the emulation technique. Let $\mathcal{G}$ be a Cayley network. The emulations emerge via the composition of two embeddings:

1. Determine a 2-embedding of $\mathcal{G}$ into $\mathcal{H} \times \mathcal{K}$ ($\mathcal{H}$ and $\mathcal{K}$ are either Cayley graphs or right coset graphs) by appealing to any one of the embedding techniques presented in this chapter.

2. Use a *projection emulation* to emulate $\mathcal{H} \times \mathcal{K}$ by either of its factors, say $\mathcal{H}$, with unit dilation, congestion $\leq |\mathcal{K}|\rho(\mathcal{G})$ and load factor $|\mathcal{K}|$. Each vertex $(u, v) \in V(\mathcal{H}) \times V(\mathcal{K})$ is assigned to vertex $u \in V(\mathcal{H})$. Edges of the form $[(u, v), (u, v')]$ in $\mathcal{H} \times \mathcal{K}$ are routed "internally" within vertex u; edges of the form $[(u, v), (u', v)]$ in $\mathcal{H} \times \mathcal{K}$ are routed via the unit-length path (u, u') in $\mathcal{H}$.

In the same fashion, we can emulate $\mathcal{H} \times \mathcal{K}$ on $\mathcal{K}$. It is easily verified that the composition of the above two embeddings preserves the processor-time product of any algorithm designed to execute on $\mathcal{G}$. Note that for many Cayley networks (in particular, those with a rich subgroup structure), we obtain a large family of possible networks, each of which can execute arbitrary algorithms of the Cayley network in a work-preserving fashion. We view this as a *processor-time* tradeoff for Cayley networks. The tradeoff is useful when the amount of hardware available is unknown during algorithm design since it allows us a considerable degree of latitude in choosing the size of the final physical network while promising efficiency.

CHAPTER 6

A TECHNIQUE FOR EMBEDDING deBRUIJN GRAPHS IN HYPERCUBES AND ITS GENERALIZATION

In this chapter, we present an embedding of $\mathcal{D}_n$ in $\mathcal{H}_n$ that has dilation at most $\lfloor 2n/5 \rfloor + 2$, and congestion 2. Previous results obtained for embedding the deBruijn graph in the hypercube appealed to heuristics, and thus are of limited use. Even though the dilation of our embedding is linear in the diameter of the hypercube (an arbitrary embedding will have dilation bounded by the host's diameter), for most realistic sizes of hypercubes that are constructed today, the result is of some practical interest. In the latter part of this chapter we generalize the embedding technique and, as a further example, we show how to embed mesh graphs in deBruijn graphs. We also suggest some other instances in which the technique may be applicable.

6.1 Related Work and Motivation

Determining the computational power of hypercubes is a central problem in the theory and practice of parallel networks, since so many parallel computers employ this topology as their underlying interconnection network. In particular, we would like to know what communication patterns the hypercube can simulate efficiently. The standard notion of an *embedding* (see Chapter 2) is the most common method for showing that one network can perform the computation of another. Towards this direction, the following embeddings of common networks in hypercubes have been discovered:

- Any binary tree can be embedded in the smallest hypercube big enough to contain it with constant dilation and load factor [27].

- Any d-dimensional mesh can be embedded in the smallest hypercube big enough to contain it with dilation $O(d)$ [36].

- Any butterfly-like graph (*i.e.*, the cube-connected cycles, the butterfly or the FFT graphs) can be embedded in the smallest hypercube big enough to contain it with dilation at most 2 and unit congestion [54] (also, see Chapter 5 of this thesis).

Along different lines, Winkler [109] has characterized the graphs that can be embedded as *subgraphs* of the hypercube, and Greenberg and Bhatt [53] have recently studied the problem of embedding *multiple copies* of the above classes of graphs in hypercubes. Before the result presented here was discovered, there were no known non-trivial embeddings of the deBruijn network in the hypercube.

A second approach to the problem of inter-simulation is through the notion of *work-preserving emulations* (defined in Section 5.2). In this model, computations of the guest network can be *replicated* at several nodes of the host network. (This may be visualized as a one-to-many embedding.) Schwabe [97] recently proved that the hypercube can emulate (in this model) any t steps of an N-node deBruijn network in $O(t)$ steps on an N-node hypercube, provided that $t \geq \log N$. However, the relationship between the two networks in a more stringent embedding model in which a one-to-one mapping is required remains unresolved. Arguably, this is the most famous unresolved question in the area of network embeddings.

6.2 An Embedding Based on Direct Product Structure

Assume for the moment that n is a multiple of 5. The embedding of $\mathcal{D}_n$ in $\mathcal{H}_n$ is based on a dilation 2 and congestion 2 embedding of $\mathcal{D}_5$ in $\mathcal{H}_5$ that was found by exhaustive computer search. (See Table 6.1.) The embedding is described in two phases:

- **Phase 1.** Embed $\mathcal{D}_n$ into a direct product of $n/5$ disjoint copies of $\mathcal{D}_5$ with dilation $n/5$.

- **Phase 2.** Decompose the hypercube $\mathcal{H}_n$ into a direct product of $n/5$ disjoint copies of $\mathcal{H}_5$. (This constitutes an isomorphism.) Embed each copy of $\mathcal{D}_5$ in a distinct copy of $\mathcal{H}_5$ contained in $\mathcal{H}_n$.

The proof that the first phase is possible is a direct consequence of the following more general fact.

Proposition 12 *For all integers $n, m > 0$ the deBruijn graph $\mathcal{D}_{n+m}$ can be embedded in the direct product $\mathcal{D}_n \times \mathcal{D}_m$ with dilation 2.*

Proof. If x is a bit string then let $[x]^k$ and $[x]_k$ denote the length-k prefix and suffix of x, respectively. Define an embedding of $\mathcal{D}_{n+m}$ in $\mathcal{D}_n \times \mathcal{D}_m$ by mapping an arbitrary vertex x to the pair $\langle [x]^n, [x]_m \rangle$. Clearly, this is a well-defined bijection between the vertex sets of $\mathcal{D}_{n+m}$ and $\mathcal{D}_n \times \mathcal{D}_m$. We can simulate an edge of $\mathcal{D}_{n+m}$ in $\mathcal{D}_n \times \mathcal{D}_m$ as follows. Let $[x]^n = \alpha y \beta$ and $[x]_n = \delta z \epsilon$, where $\alpha, \beta, \delta, \epsilon \in \{0, 1\}$. We can simulate any "shift" of x (*i.e.*, any edge of $\mathcal{D}_{n+m}$ that has x as one of its endpoints) via 2 independent "shift" operations on $[x]^n$ and $[x]_m$ since $(\alpha y \beta, y \beta \delta)$ and $(\delta z \epsilon, z \epsilon \gamma)$ are edges of $\mathcal{D}_n$ and $\mathcal{D}_m$, respectively, for $\gamma \in \{0, 1\}$. Hence, the dilation of the embedding is transparent. □

Applying this proposition inductively, the first phase can be realized with dilation $n/5$; we repeatedly "peel off" copies of $\mathcal{D}_5$, increasing the dilation by 1 for each copy. The second phase is easily accomplished since, by definition of the graph-theoretic direct product, the hypercube $\mathcal{H}_n$ is isomorphic to the direct product of $n/5$ copies of $\mathcal{H}_5$.

It remains to define the routing function of the embedding and analyze its congestion. To this end, suppose ϕ_1 and ϕ_2 are (one-to-one) embeddings of $\mathcal{D}_n$ in $\mathcal{H}_n$ and $\mathcal{D}_m$ in $\mathcal{H}_m$,

respectively. Consider the new embedding, ϕ^*, of $\mathcal{D}_{n+m}$ in $\mathcal{H}_{n+m}$ given by

$$\phi^*(z) = \langle \phi_1([z]^n), \phi_2([z]_n) \rangle,$$

where z is a vertex of $\mathcal{D}_{n+m}$ (*i.e.*, a bit string of length $n+m$). Suppose (z, z') is an edge of $\mathcal{D}_{n+m}$ where $z = xy$, $z' = x'y'$, the length of x is n and the length of y is m. (Both x' and y' depend on whether (z, z') is a shuffle or a shuffle-exchange edge.) Note that (x, x') is an edge of $\mathcal{D}_n$ and (y, y') is an edge of $\mathcal{D}_m$ (see Proposition 12). We obtain a path between $\phi^*(z)$ and $\phi^*(z')$ by concatenating the following two paths in $\mathcal{H}_{n+m}$:

1. The first is a path between $\langle \phi_1(x), \phi_2(y) \rangle$ and $\langle \phi_1(x'), \phi_2(y) \rangle$ (fixing the second coordinate).

2. The second is a path between $\langle \phi_1(x'), \phi_2(y) \rangle$ and $\langle \phi_1(x'), \phi_2(y') \rangle$ (fixing the first coordinate).

It now follows that $\mathsf{cong}(\phi^*) = \max\{\mathsf{cong}(\phi_1), \mathsf{cong}(\phi_2), 2\}$ since ϕ^* is a one-to-one embedding and the guest graph is a direct product. (This argument is similar to that presented for the Schreier embedding technique in Chapter 5 – see Theorem 13.) The reason that the congestion is at least 2 (independent of the embeddings of each factor) is that a de-Bruijn graph of order 1 consists of a single edge (across which 2 messages may travel). It follows by induction that the total congestion of the embedding is 2, since the congestion of the embedding of $\mathcal{D}_5$ in $\mathcal{H}_5$ is 2. In summary, we have the following theorem.

Theorem 16 *The order-n deBruijn graph can be embedded in the order-n hypercube with congestion* 2 *and dilation*

$$\mathsf{dil}(\mathcal{D}_n \rightarrow \mathcal{H}_n) = \begin{cases} 2n/5 & n \equiv 0 \bmod 5 \\ \lfloor 2n/5 \rfloor + 1 & n \equiv 1 \bmod 5 \\ \lfloor 2n/5 \rfloor + 2 & \textit{otherwise} \end{cases}$$

Proof. For arbitrary n, express n as $m+r$, where m is a multiple of 5 and $0 \leq r \leq 4$. By appealing to Proposition 12, we embed $\mathcal{D}_n$ in a direct product of $\mathcal{D}_m$ and $\mathcal{D}_r$ with dilation

$\mathcal{H}_2$	$\mathcal{D}_2$
0	0
1	1
2	2
3	3

$\mathcal{H}_3$	$\mathcal{D}_3$
0	0
1	1
2	4
3	2
4	7
5	3
6	6
7	5

$\mathcal{H}_4$	$\mathcal{D}_4$
0	3
1	1
2	9
3	8
4	2
5	0
6	4
7	10
8	6
9	7
10	12
11	14
12	11
13	5
14	13
15	15

$\mathcal{H}_5$	$\mathcal{D}_5$
0	0
1	1
2	2
3	3
4	4
5	6
6	5
7	7
8	8
9	12
10	10
11	17
12	9
13	14
14	15
15	19
16	16
17	26
18	21
19	27
20	18
21	29
22	30
23	13
24	20
25	24
26	31
27	22
28	28
29	25
30	11
31	23

Table 6.1: Embeddings of small deBruijn graphs in hypercubes

2. It follows from the results of a computer search (see Table 6.1) that for $2 \leq n \leq 4$, $\mathcal{D}_n$ can be embedded in $\mathcal{H}_n$ with dilation 2 and congestion 2. We now proceed by following the above method to obtain a dilation $2m/5 + 2$, congestion 2 embedding of $\mathcal{D}_n$ in $\mathcal{H}_n$. The theorem follows. □

6.3 A Generalized Embedding Technique Based on Direct Products

Notice that the embedding we have presented hinges only on two things. Namely, the existence of a "factorization" of the deBruijn graph into a direct product of smaller "factors", and that each of the factors can be efficiently embedded into smaller hypercubes. This suggests the following generalized approach for attacking a wide variety of embedding problems:

Strategy for Embedding $\mathcal{G}$ into $\mathcal{H}$:

- **Step 1.** Embed $\mathcal{G}$ into a non-trivial direct product graph, $\prod \mathcal{F}_i$.
- **Step 2.** Find a collection of graphs $\{\mathcal{H}_j\}$ such that $\prod \mathcal{H}_j$ can be efficiently embedded in $\mathcal{H}$.
- **Step 3.** Embed each factor $\mathcal{F}_i$ into one of the factors $\mathcal{H}_j$.

The final embedding of $\mathcal{G}$ into $\mathcal{H}$ is formed by composing the embeddings in Steps 1, 3 and 2, in that order. We remark that an instantiation of this scheme was employed in [10] for embedding the Butterfly graph in the deBruijn graph with sub-logarithmic dilation, although the general nature of the technique was not recognized there. In the following section, we further exemplify the technique by determining embeddings of mesh graphs into deBruijn graphs.

6.3.1 Embedding Mesh Graphs in deBruijn Graphs

Let $\mathcal{M}_{n,m}$ denote an $n \times m$ mesh graph (having n rows and m columns). It is obvious that

$$\mathcal{M}_{n,m} = \mathcal{L}_n \times \mathcal{L}_m.$$

Since the deBruijn graph is Hamiltonian, $\mathcal{L}_k$ (for any value of k) is a subgraph of any deBruijn graph large enough to contain it. It follows that $\mathcal{L}_n \times \mathcal{L}_m$ is a subgraph of

$\mathcal{D}_{\lceil \log n \rceil} \times \mathcal{D}_{\lceil \log m \rceil}$. Our remaining task is to embed $\mathcal{D}_{\lceil \log n \rceil} \times \mathcal{D}_{\lceil \log m \rceil}$ in $\mathcal{D}_{\lceil \log n \rceil + \lceil \log m \rceil}$. To achieve this we appeal to a result of [10] which analyzed the properties of the straightforward embedding that assigns a node in the domain, say (u, v), to the concatenation of u and v. The dilation of this embedding was found to be

$$O(\min\{\lceil \log n \rceil, \lceil \log m \rceil\})$$

with dynamic congestion $O(1)$. We remark that this embedding is poor for "nearly square" meshes. However, for meshes in which the length of one of the sides is logarithmic in the other side, we achieve an *exponential* speedup over an arbitrary embedding.

Extending this embedding to the *toroidal* mesh graph is simple, since the deBruijn graph is pancyclic [113] (*i.e.*, it contains a cycle of every length). This implies that $\mathcal{R}_n \times \mathcal{R}_m$ is a subgraph of $\mathcal{D}_{\lceil \log n \rceil} \times \mathcal{D}_{\lceil \log m \rceil}$, and thus the same embedding applies. We summarize our results in the following theorem.

Theorem 17 *The $n \times m$ (toroidal) mesh graph can be embedded in the deBruijn graph with expansion at most 2, dilation at most*

$$O(\min\{\lceil \log n \rceil, \lceil \log m \rceil\})$$

and constant dynamic congestion.

We mention a final area in which we believe our embedding technique may yield new results. Since higher-order deBruijn networks can be decomposed into direct products (in an analagous way to that embodied in Proposition 12), we suspect that the embedding technique will yield non-trivial embeddings of the higher-order deBruijn networks in hypercubes. The success of this technique demands that we first find efficient embeddings of small versions of these networks into appropriately sized hypercubes.

CHAPTER 7

A Deterministic Reconfiguration Strategy for Faulty Hypercubes

This chapter presents and analyzes a deterministic algorithm for reconfiguring a hypercube in the presence of faults. In effect, the algorithm provides for a simulation of a fault-free machine (of the original size) with only a small penalty in communication delay. It is based on a decomposition of the N-vertex hypercube into $\Theta(\log N)$-size disjoint *stars* that arises from the theory of error-correcting codes. More precisely, we describe an embedding of a fault-free network in a faulty one that has dilation 5, unit dynamic congestion and a load factor that is at most $\lceil 6/\alpha \rceil + 1$, where α is the smallest fraction of vertices that are live (*i.e.*, non-faulty) in a single star (see Section 7.2 for detailed definitions). Our fault model assumes that only the *computational elements* fail and that it is possible to route messages through faulty processors.

The algorithm that produces the embedding takes deterministic time $O(N^2 \log N)$ in an off-line sequential setting and is the first *deterministic* solution to this problem.

7.1 Introduction and Related Work

As we have discussed elsewhere in this thesis (see the Introduction and Chapter 4), an important consideration in the design of a large-scale parallel network is the degree to which it is resilient to faulty components. A method for coping with faults in the network is imperative, and in Chapter 4, we discussed two different approaches for dealing with this problem. Both of these approaches examined the ability of a network to *route* (either permutation routing or point-to-point routing) in the presence of faulty PEs. An

alternative approach to this problem is to *reconfigure* the network (via software) so that the remaining non-faulty part of the network can simulate a fault-free network efficiently on *arbitrary computations*. This is the approach taken in several papers on the subject [8,32,37,57,58,73] and is the approach adopted in this work.

In essence, our salvaging algorithm is a deterministic analogue of a result of Hastad, Leighton and Newman [58] under a more restrictive fault model. The main result of the work of Hastad, Leighton and Newman is a randomized algorithm for embedding a fault-free hypercube in a faulty one implying a constant delay simulation with high probability, assuming that the vertices fail randomly and independently with fixed probability $p < 1$. Thus, to within constants, this result is optimal and implies an extremely strong degree of fault-tolerance for hypercube networks. An important difference between the fault model used in this work compared to that in [58] is that we allow messages to be routed through faulty vertices, even though the faulty vertices have lost their *computational* capability. This assumption is justified by the fact that processing elements usually involve much more complicated components rendering them more prone to failure than switching elements.

Relationship to Constant Distance Permutation Routing. In work of Annexstein [8], efficient algorithms for salvaging the deBruijn and butterfly networks (with slowdown $O(\log \log N)$ for an N-vertex network) using the same fault model as the one in this work, were presented. Annexstein's general approach is to partition the vertex set of a network into disjoint sets, each of size $\Theta(\log N)$, and then use (off-line) permutation routing to find an appropriate assignment of dead vertices to live vertices with non-congesting paths between them. Unfortunately, in the case of the hypercube, this approach is not directly applicable since it is an open question whether a permutation in which the destination of a message is a constant distance away from the message's origin can be routed in a constant number of steps. Thus, a new method for performing the assignment of

dead vertices to live ones must be developed. In a sense, our algorithm generalizes Annexstein's reconfiguration strategy, in that we are using a similar assignment strategy with the important difference that we do not appeal to permutation routing to find the assignment.

Salvaging Algorithms and Fault-Distribution Assumptions. Our algorithm deals with worst-case faults in the following "localized" sense: The performance of the algorithm depends only on the maximum number of faulty vertices that exists in any single $\Theta(\log N)$-size star. Thus, the algorithm performs most efficiently when there is an even distribution of faults among the stars – internally, in each star, the exact fault pattern does not affect the performance of the algorithm. In contrast to this scenario, if we consider arbitrary worst-case fault patterns, then in order to guarantee a constant factor slowdown, the N-vertex hypercube can have at most $O(\log N)$ faults [20,73]. Other recent results [32] show that, in the worst case, a restricted class of normal algorithms (see [103]) can be run with constant slowdown given a polylogarithmic number of faults. The algorithm presented in this chapter occupies a "middle ground" between handling worst-case faults and algorithmic generality: Firstly, as long as faults are distributed evenly among the stars, the resulting simulation is efficient; and secondly, the simulation makes no assumptions about the algorithms that will be executed.

The remainder of this chapter is organized as follows. Section 7.2 describes the fault model under consideration and outlines our approach for constructing the embedding. Section 7.3 presents the assignment algorithm and proves its correctness. Finally, Section 7.4 discusses the implementation of the algorithm.

7.2 The Fault Model and the Approach

Consider the 2^n-node hypercube, $\mathcal{H}_n$. If s is a vertex of $\mathcal{H}_n$ then let s^i denote the neighbour of s obtained by flipping the i-th bit of s. The vertex s^i is said to be **in the**

i-th dimension of s.

Each vertex is assumed to have a *processing element* that handles computation and a *switching element* that handles inter-processor communication. For example, the architecture of the connection machine is designed this way [60]. We assume the *single-port* model of communication. As previously mentioned, we only consider faults involving processing elements, so that the underlying communication network remains intact.

Let $\mathcal{H}'_n$ denote a hypercube in which some of the vertices have failed. The salvaging operation is accomplished via an *embedding* of $\mathcal{H}_n$ into $\mathcal{H}'_n$. First, we partition the vertices of $\mathcal{H}'_n$ into disjoint sets and find an appropriate *assignment* of dead vertices to live vertices within each set[1]. Initially, we restrict ourselves to hypercubes where $n = 2^r - 1$ for some $r > 0$. (Later, we extend this to arbitrary n – see Section 7.3.4.) In this case, it is a well-known fact from the theory of error-correcting codes (see, for example [84]) that the vertices of $\mathcal{H}_n$ can be partitioned into $2^n/(n+1)$ **stars**. Each star consists of a **center** vertex and its n adjacent vertices. The centers of the stars comprise a perfect single-error correcting code. In effect, this means that any length n bit string is either a center vertex (also called a codeword) of some star or is adjacent to a *unique* center vertex of some star. Hence, any codeword that has at most a single error in it can be uniquely "decoded". In purely graph-theoretic terms, the centers of the stars comprise a *dominating set.*

Notation. We denote all of the vertices in a star centered at s by star(s).

The assignment algorithm is the crux of the embedding (see the next section). Once it has been made, the routing of an edge (u, v) in $\mathcal{H}_n$ is accomplished in $\mathcal{H}'_n$ in three steps. Let L_u and L_v be the live vertices assigned to u and v in $\mathcal{H}'_n$, respectively.

1. Route from L_u to u.

[1]Live vertices are, by default, mapped to themselves.

2. Route along the edge (u, v).

3. Route from v to L_v.

The paths chosen to connect live and dead vertices will be of length 2, and thus the dilation is bounded above by 5. This routing technique is analogous to the one employed in [8] for salvaging the deBruijn and butterfly networks.

7.3 The Assignment Algorithm

Suppose there are at least αn live vertices in each star, for some constant $0 < \alpha \leq 1$. We do not consider the center vertices of stars since, if they are dead, we simply assign them to any live vertex in the star increasing the load factor by at most 1.

Let s be the center vertex of some star of $\mathcal{H}'_n$. Our aim is now to assign live nodes of star(s) to dead vertices of star(s) with the following two requirements:

1. The load is as balanced as possible.

2. The number of other assignments using the vertex s^{lj} is minimized.

Let s^l be a live vertex in star(s), and let s^d be a dead vertex in star(s). We shall always use a *pair* of adjacent edges to connect s^l and s^d. Namely, we use the edge (s^l, s^{ld}) followed by the edge (s^{ld}, s^d). The key point to notice about this choice is that it avoids the obvious congestion that would occur if we had chosen to route through the center vertex of the star. By definition, we call the vertex s^{ld} an **intermediate vertex.**

An Overview of the Algorithm. The algorithm proceeds in a greedy fashion in conjunction with backtracking. By this, we mean the following. Let us focus on a particular star, say star(s). Suppose we are trying to find a live vertex in star(s) to assign to some dead vertex in star(s). We may consider using any of the live vertices in star(s) such that the appropriate intermediate vertex has not been claimed by another star (for

some other assignment). If this is the case, then we have already succeeded (we go ahead and assign it). If, on the other hand, all of the intermediate vertices in question have already been claimed by other stars, then we see if it is possible to "free up" one of these intermediate vertices by changing some assignment for one of these new stars. Assuming that this is possible, say by changing an assignment in star(s'), then we can again succeed since, after changing the assignment for star(s'), we have freed up an intermediate vertex that can be used by star(s). If it is not possible, then we repeat the procedure recursively. The remainder of this section shows that this procedure succeeds as long as there is a sufficient number of live vertices available in each star.

Let β be a fraction that depends only on α. We define this value precisely in the next section. Divide the n dimensions into $c = \lceil \frac{1}{\beta} \rceil$ disjoint blocks, each of size $\lfloor \beta n \rfloor$ (except for possibly the last which may contain less than $\lfloor \beta n \rfloor$ dimensions). Denote the blocks by

$$B_1, B_2, \cdots, B_c.$$

The algorithm processes each B_i ($1 \leq i \leq c$) separately. By this we mean that the algorithm finds an assignment for all of the dead vertices that are in a dimension in block B_i independently of the other blocks.

Terminology. Assume the algorithm is busy processing the dimensions in block B_i.

1. A live vertex is said to be **participating** if it has not yet been assigned a dead vertex in B_i.

2. A non-center vertex is **available** if it has not yet been utilized as an intermediate vertex while processing B_i.

3. Once a participating vertex is assigned to some dead vertex it becomes **non-participating** and the intermediate vertex involved in this assignment becomes **not-available.**

The algorithm is outlined below.

Algorithm Salvage()

- For each block B_k do the following:

  1. Mark all **live** vertices as **participating.**
  2. Mark all **non-center** vertices as **available.**
  3. For each dimension $j \in B_k$ and for each star star(s) whose j-th dimension is dead, do the following:
     (a) Assign some **participating** vertex v in star(s), whose **intermediate** vertex v^j is available, to s^j. Note that this step may require *backtracking.*
     (b) Mark v as **non-participating** and v^j as **not-available.**

Analysis. Once we have proved the correctness of the algorithm, the claimed load factor and congestion bounds will follow directly: Since the outer loop repeats c times (once for each block of dimensions), and for each block of dimensions the assignment found by the algorithm has a load factor of one and unit congestion, the load factor accrued during the entire algorithm cannot exceed c. If we simulate the blocks sequentially, the dynamic congestion is equal to 1, since the assignments made by the algorithm for a single block incur unit congestion. This is important since no extra hardware in the form of queues is required to implement the simulation.

7.3.1 Alternating Trees

The difficult part of the algorithm is finding an appropriate participating vertex to assign to s^d, *i.e.*, we must find a participating vertex whose intermediate vertex is available. In general, this may not be immediately possible. We may have to backtrack and reassign some previous assignments in order to achieve this. To this end, we inductively define a

labeled rooted tree, the purpose of which is to construct a sequence of vertex sets,

$$V(0),\ V(1),\ldots,$$

representing the set of new intermediate vertices as we search outwards from star(s).

We call the tree an **alternating tree**[2] and denote it by T_s^d. The root of T_s^d denotes star(s). For the sake of exposition, we let each level of the tree consists of two **tiers**. The first level is constructed as follows:

> **Tier 1 edges of level 1.** For each edge in the d-th dimension that emanates from a participating vertex of star(s), we form an edge of T_s^d and label it with d. The new vertices formed by these edges, denoted by $V(0)$, represent the set of *distinct* possible intermediate vertices.

The reason these intermediate vertices are distinct is because they are all in the d-th dimension[3]. If one of these vertices is available then the tree terminates. Otherwise, we construct the second tier of the first level:

> **Tier 2 edges of level 1.** For each intermediate vertex v, we add an edge between v and a new vertex representing the star which is already using v as an intermediate vertex. Label these new edges by their dimension.

Suppose we have constructed the i-th level of T_s^d ($i \geq 0$). Let $V(i)$ be the set of distinct intermediate vertices in this level. We proceed to construct the next level of T_s^d.

> **Tier 1 edges of level $i+1$.** Suppose (v', s') is a second tier edge of level i labeled by j. Consider all of the possible *alternative* intermediate vertices for star(s') in the j-th dimension, using the set of all participating vertices

[2]This term is derived from the use of the word "alternating" in the theory of graph matchings, and should not be confused with the corresponding term found in the theory of alternating Turing machines.

[3]The set of all edges in a fixed dimension in a hypercube forms a perfect matching.

in star(s'). For each alternative that has not been encountered before in the construction of the tree we add a new edge from the vertex s' to the new vertex w (representing the new possible intermediate vertex). Repeat this for each second tier edge of level i. The set of new intermediate vertices formed by this process is denoted by $V(i+1)$.

If any of these vertices is available then the tree terminates. Otherwise, we proceed to construct the second tier of level $i+1$.

Tier 2 edges of level $i+1$. For each intermediate vertex, we add a single new edge to vertex $w \in V(i+1)$, say (w, s''), labeled by l, if the vertex w has already been assigned as an intermediate vertex in the l-th dimension of star(s'').

An **alternating path** is defined to be a path in an alternating tree originating at the root. An alternating path is represented by a sequence of 5-tuples each of which represents the two edges and three vertices involved as the path moves through a single level, *e.g.*, the 5-tuple (s_1, j_1, I, j_2, s_2) denotes the first tier edge (s_1, I) labeled by j_1 followed by the second tier edge (I, s_2) labeled by j_2. The reader may find it useful to think of the above 5-tuple as follows:

Star(s_1) is requesting I as an intermediate vertex in order to assign some participating vertex to the dead vertex $s_1^{j_1}$. But, star(s_2) has already claimed I as an intermediate vertex in order to satisfy the dead vertex $s_2^{j_2}$.

Finding an alternating path in T_s^d that terminates at an available vertex enables us to assign a participating vertex to s^d (after a reassignment of the other vertices along the alternating path) as the following lemma shows.

Lemma 6 *If there is an alternating path in T_s^d terminating at an available vertex then, by rearranging the assignments along the path, the d-th dimension of s can be assigned a participating vertex.*

Proof. Suppose

$$(s, d, I_1, j_1, s_1), (s_1, j_1, I_2, j_2, s_2), \cdots, (s_{m-1}, j_{m-1}, I_m, j_m, s_m), (s_m, j_m, I_{m+1})$$

is an alternating path terminating at the available vertex I_{m+1}. Reassign the j_m-th dimension of s_m using the (available) intermediate vertex I_{m+1}. This reassignment removes the constraint from the previous star in the path, *i.e.*, I_m is no longer requested as an intermediate vertex. Thus, we may reassign the j_{m-1}-st dimension of s_{m-1} using the intermediate vertex I_m. This removes the constraint from the intermediate vertex I_{m-1}. Continuing in this fashion, we eventually assign the d-th dimension of s using the intermediate vertex I_1. □

7.3.2 Proof that Alternating Trees Grow

Observe that if an alternating tree grows monotonically then eventually it must contain an available vertex since $\mathcal{H}_n$ is finite. We determine, by induction, the relationship between α and β that will ensure this occurs[4]. This, in turn, will guarantee (by Lemma 6) that the solution can proceed, *i.e.*, the current dimension of the star at the root of the alternating tree can be satisfied.

In order to make the argument precise, suppose for the moment that there are at least n_ℓ live vertices in each star and we have found an assignment for n_a dimensions in some stars and $n_a + 1$ dimensions in the remainder. Suppose the $(n_a + 1)$-st dimension is m. Let star(s) be a star that has only n_a dimensions assigned. Form the alternating tree T_s^m. We now prove by induction that $|V(i)| \geq 2|V(i-1)|$ as long as $n_a \leq n_\ell/6$.

[4]Actually, we require the tree to grow geometrically – this is useful (see Section 7.4) for finding an efficient implementation of the algorithm.

BASIS: The number of vertices in level 0 of T_s^m is

$$V(0) = n_\ell - n_a.$$

Now consider the second level. The minimum size of $V(1)$ is obtained by taking the total number of distinct *edges* that can arise from $V(0)$ minus the maximum number of these edges that have an endpoint in $V(0)$, and dividing this quantity by the maximum number of edges that can share a single vertex. We discuss each of these quantities in turn: The first quantity is at least

$$(n_\ell - n_a - 1)(n_\ell - n_a),$$

since a star that has satisfied $n_a + 1$ dimensions will still have at least $n_\ell - n_a - 1$ participating vertices. The second quantity is bounded above by

$$(n_a - 1)(n_\ell - n_a),$$

since for a given vertex $v \in V(0)$, there are at most $n_a - 1$ new edges that can be used to satisfy some assignment (by definition of T_s^m, two edges incident to v have already been used). Finally, because there are at most n_a+1 dimensions that have been satisfied, there are at most n_a+1 possible requests for a given intermediate vertex. But, because of our assumption that all intermediate vertices are not available, this quantity is bounded above by n_a.

Hence, we have

$$\begin{aligned} |V(1)| &\geq \frac{(n_\ell - n_a - 1)(n_\ell - n_a) - (n_a - 1)(n_\ell - n_a)}{n_a} \\ &\geq \frac{(n_\ell - n_a - 1)(n_\ell - n_a)}{n_a} - (n_\ell - n_a) \end{aligned}$$

To guarantee that $|V(1)| \geq 2|V(0)|$, it suffices to have

$$(n_\ell - n_a - 1) \geq 3n_a$$

and hence

$$n_a \leq \frac{n_\ell}{4} - \frac{1}{4}.$$

INDUCTIVE STEP: Now suppose that for $k \leq i-1$ we have $|V(k)| \geq 2|V(k-1)|$ and that there are no available vertices on level i. The proof that $|V(i)| \geq 2|V(i-1)|$ uses the same counting argument (along with the inductive hypothesis) contained in the basis to obtain a lower bound on the number of vertices in $V(i)$ in terms of $V(i-1)$. By analogy with the basis, we have

$$\begin{aligned} |V(i)| &\geq \frac{(n_\ell - n_a - 1)\,V(i-1) - (n_a - 1)\sum_{k=0}^{i-1}|V(k)|}{n_a} \\ &\geq \frac{n_\ell - n_a - 2}{n_a - 1}|V(i-1)| - 2|V(i-1)| \end{aligned}$$

The last inequality follows by the inductive hypothesis: Since the sequence $\{|V(k)|\}_{k=0}^{i-1}$ is geometric it is evident that

$$\sum_{k=0}^{i-1}|V(k)| \leq 2|V(i-1)|.$$

In order to satisfy $|V(i)| \geq 2|V(i-1)|$ it suffices to have

$$\frac{n_\ell - n_a - 1}{n_a} \geq 4.$$

Simplifying this we obtain

$$n_a \leq \frac{n_\ell}{5} - \frac{1}{5}.$$

Finally, by combining the conditions on n_a required by both the basis and the inductive step, we infer that there exists at least one available vertex in the alternating tree if

$$n_a \leq \min\{\frac{n_\ell}{4} - \frac{1}{4}, \frac{n_\ell}{5} - \frac{1}{5}\},$$

which is clearly satisfied[5] if

$$n_a \leq \frac{n_\ell}{6}.$$

We now express our result in terms of fractions. Let $n_\ell = \alpha n$ and $n_a = \beta n$. By substituting these values into the above equation, we obtain

$$\beta \leq \frac{\alpha}{6}$$

[5]We may assume that $n_\ell > 1$, for otherwise, the solitary live vertex will simulate *all* remaining vertices in each star.

This completes the proof that alternating trees grow when β is appropriately chosen. □

7.3.3 An Annotated Example

To give the reader a feel for how the salvaging algorithm works, this section presents an annotated example. Suppose we are in the middle of the assignment algorithm and we are trying to find an assignment for a dead node in the m-th dimension of star(s) (refer to Figure 7.1). We first examine the availability of the intermediate vertices of star(s) using all of the participating vertices of star(s). In Figure 7.1, this set is equal to $\{I_1, I_2, I_3\}$. For each intermediate vertex, the parameter in parentheses represents the star which has already claimed it (at some previous stage of the algorithm). Since all of these intermediate vertices are already claimed, we repeat the process (only partially shown in Figure 7.1 for clarity). Notice that the intermediate vertex I_8 is available. The corresponding alternating tree is depicted in Figure 7.2. By reassigning the assignments made along the path from the root to I_8, we have achieved our aim of assigning the original dead node in star(s). The state of the hypercube after the reassignments have been made is depicted at the bottom of Figure 7.1.

A Remark on the Algorithm's Performance. As a general indication of the performance of the salvaging algorithm, we observe that as long as more than 6/7 of the vertices in each star are live, the resultant embedding has dilation 5, load factor bounded above by 7 and unit dynamic congestion. It is interesting to note that there is no improvement in the efficiency of the embedding if the live fraction of vertices in each star is further increased.

7.3.4 Extending the Algorithm to the General Case

Let n be a natural number. Choose r such that

$$2^r - 1 \leq n < 2^{r+1} - 1$$

and let $d = 2^r - 1$. Then, since

$$\mathcal{H}_n = H_{n-d} \times H_d,$$

$\mathcal{H}_n$ can be decomposed into 2^{n-d} copies of H_d. Notice that $d \geq n/2$. Define an **altered star** to be a single star in one of the copies of H_d. The salvaging algorithm can now be applied to each copy of H_d independently.

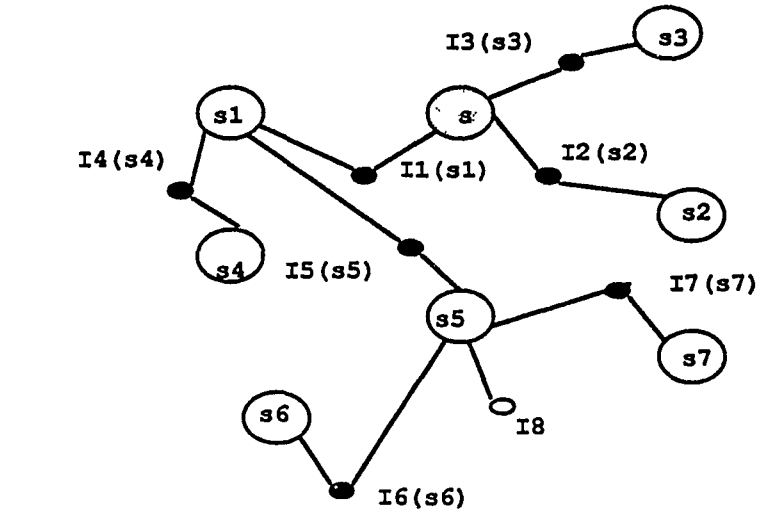
7.4 Implementing the Algorithm

In this section, we discuss implementation issues regarding the salvaging algorithm. Suppose we have a representation of the faulty hypercube as a graph in which dead vertices are "marked". Order the vertices of each star s so that the i-th dimension of s is the i-th vertex in the ordering. Whenever there is a choice of assignments for some star, we choose the one with the lowest order. The off-line implementation of this algorithm requires us to find alternating paths efficiently. A depth-first search of the alternating tree is sufficient for this. Since there are $N \log N$ edges in the hypercube this takes at most $O(N \log N)$ steps. Hence, since there are $\log N$ dimensions and $O(N/\log N)$ stars, the total number of steps required to determine the embedding is $O(N^2 \log N)$.

We convert this algorithm to one which runs on the hypercube using only local control by observing that an alternating tree has $O(\log N)$ levels since it grows geometrically. Searching this tree in parallel can be done easily in $O(\log^2 N)$ communication steps using the single-port model of communication. Now suppose that we allow each star to try and satisfy its i-th dimension in parallel. If two alternating trees happen to overlap then we allow one to continue its search while the other is temporarily suspended. As soon as the tree that proceeded succeeds in finding a free vertex the suspended alternating tree may resume its search. In the worst case, this algorithm becomes linear in the size of the hypercube, giving a running time of $O(N \log^2 N)$. However, for most fault patterns we expect that the depth of a given alternating tree will be small.

We summarize this chapter in the following theorem.

Theorem 18 *Suppose $\mathcal{H}'_n$ is a faulty hypercube in which at least αn vertices are live in each altered star, for some constant positive fraction α. Then there is an embedding of $\mathcal{H}_n$ into $\mathcal{H}'_n$ that has dilation 5, load factor at most $\lceil 6/\alpha \rceil + 1$ and unit dynamic congestion. Furthermore, there is a deterministic procedure for finding the embedding whose time complexity is polynomial in the size of the hypercube.*



O = AVAILABLE INTERMEDIATE VERTEX

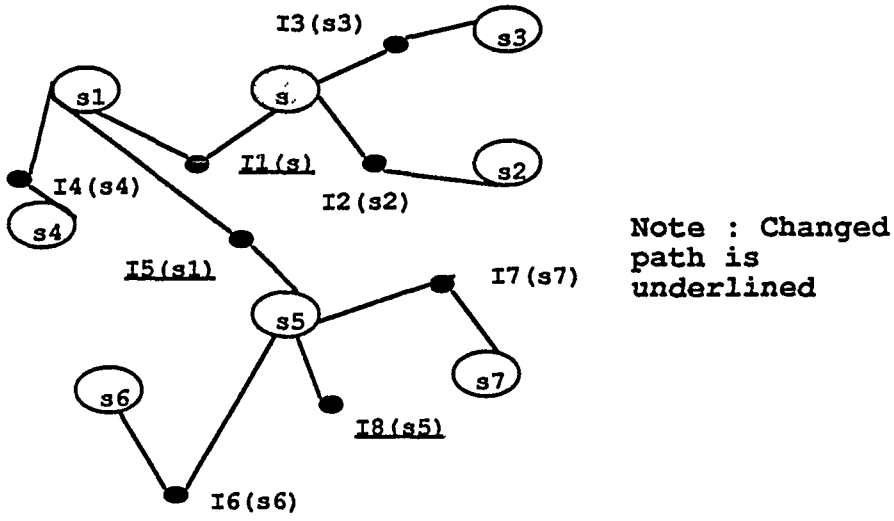


Figure 7.1: Two snapshots of a hypercube during execution of the salvaging algorithm

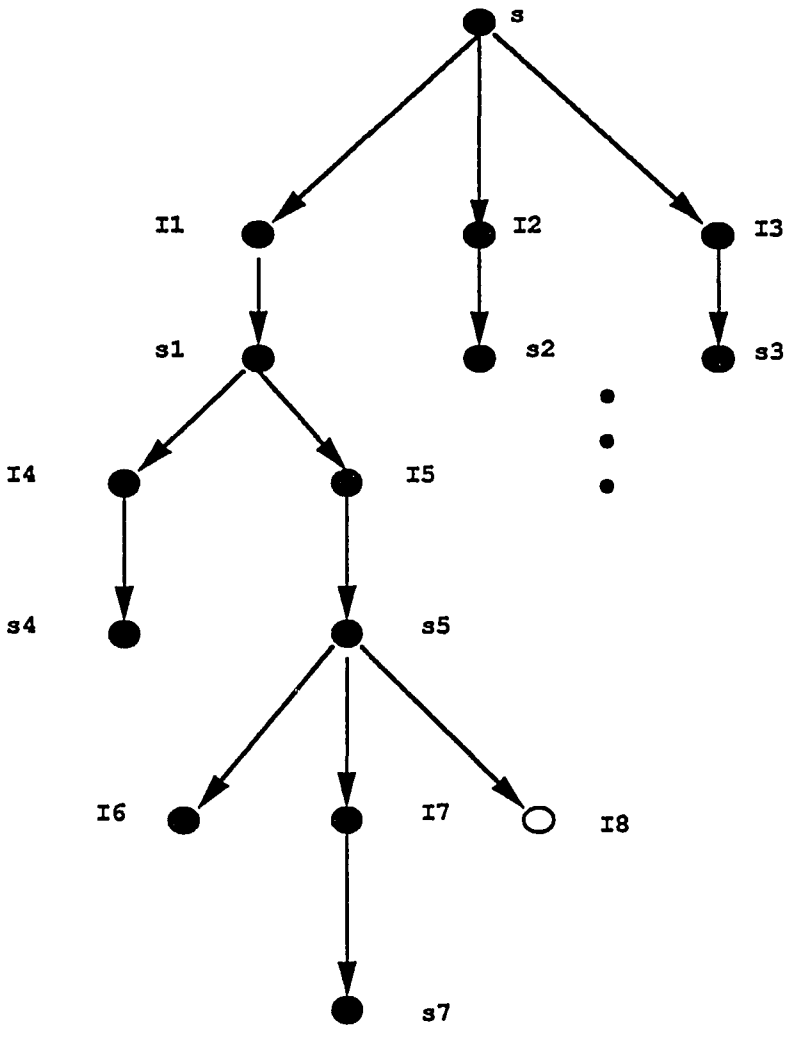


Figure 7.2: The alternating tree corresponding to Figure 7.1

CHAPTER 8

HAMILTONIAN GENERATING SETS FOR WREATH PRODUCTS

A **Hamiltonian generating set** for a finite group G is defined to be a generating set S for G with the following properties: S is minimal and the Cayley digraph $\vec{\Gamma}(G, S)$ has a (directed) Hamiltonian circuit. It has been conjectured by Klerlein [68] that all groups possess such a set, although this conjecture is far from resolved. In this chapter, we determine a Hamiltonian generating set for the wreath product of two cyclic groups and extend this result to an *iterated* wreath product of cyclic groups. We also discuss some related problems.

Computational Motivations. The Cayley graphs discussed in this chapter are a generalization of the *butterfly* graphs that we have focused on throughout this thesis. They play an important role in the theory of parallel computation. In particular, their adjacency pattern reflects the movement of data during a parallel discrete fourier transform computation (see [103]). Also, these graphs have been proposed and constructed as interconnection networks for general-purpose parallel computers [13,19]. In general, the existence of Hamiltonian circuits in the underlying graph of a parallel network has applications in data structure problems [41], and can be used as a tool in algorithm design [35,54] and in fault-tolerance [85].

Are All Cayley Graphs Hamiltonian? An excellent survey of results in the area of Hamiltonian circuits in Cayley graphs is presented in a paper by Witte and Gallian [110]. Here, we will only summarize some relevant results. An important point is to distinguish between the undirected case and the directed case. In the sequel, we use the term "cycle"

and "circuit", respectively, to differentiate between the two cases. For the former, it has been conjectured by several researchers that every Cayley graph possesses a Hamiltonian cycle, and, to a large extent this question remains unresolved. The most general result confirms that every Cayley graph with a cyclic commutator subgroup of prime-power order has a Hamiltonian cycle. Open subproblems abound in this area. For example, the problem for the dihedral groups has not yet been solved for all generating sets (see [7]).

On the other hand, in the directed case, it is known that not all Cayley digraphs have Hamiltonian circuits. For example, there is a large family of Cayley graphs of direct products of cyclic groups with the standard generating set known not to possess Hamiltonian circuits [101]. However, it has been conjectured by Klerlein [68] that *every* group has a minimal generating set for which the resultant Cayley digraph is Hamiltonian. Strikingly, it is not even known whether this is true for the class of solvable groups (see [110]).

Wreath Products. A fair amount of attention has been devoted to studying the wreath product of cyclic groups. For the undirected case, Witte and Gallian posed the following research problem [110] (Problem (2), Section 5.6): When is there a Hamiltonian cycle in the Cayley graph of the wreath product of two cyclic groups with the standard generating set? This question has only been partially answered. Stong [100] has shown that the Cayley graph of $Z_m \wr Z_n$ with the standard generating set has a Hamiltonian cycle if m is even or equal to 3. The case when m is odd and not equal to 3 remains open.

In the directed case, it is well-known that the Cayley digraph of $Z_2 \wr Z_3$ with the standard generating set *does not* have a Hamiltonian circuit. Thus, it is necessary to look for a different generating set to verify Klerlein's conjecture for wreath products. Our result exposes such a set and we generalize it to produce a Hamiltonian generating set for an arbitrary iterated wreath product of cyclic groups and for a semidirect products of cyclic groups.

8.1 The Basic Construction for Wreath Products

Suppose G is generated by $S = \{s_1, s_2, \ldots, s_k\}$. Let $Z_n = \{0, 1, \ldots, n-1\}$ be the group of integers modulo n, written additively. Let $a = 1 \in Z_n$ so that a generates Z_n. Recall the definition of the wreath product of G by Z_n, denoted $G \wr Z_n$. Let $G^n = G \times G \times \cdots \times G$, the direct product of n copies of G. G^n is called the **base group** of $G \wr Z_n$. The natural action of a on an element of the base group $(g_1, g_2, \ldots, g_n)$ gives $(g_2, g_3, \ldots, g_1)$, *i.e.*, a cyclically permutes the factors of the base. This extends to an action of Z_n on G^n and hence we can form the semidirect product of G^n by Z_n, which is, by definition, the wreath product $G \wr Z_n$. An element of the wreath product will be written as a pair $\langle x; y \rangle$ where $x \in Z_n$ and $y \in G^n$. We refer to x as the **first component** and y as the **second component.**

Consider the counting generating set, $\mathbf{C}$, for $G \wr Z_n$. It suffices to restrict our attention to $Z_m \wr Z_n$ since $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$ is a subgraph of $\vec{\Gamma}(G \wr Z_n, \mathbf{C})$ when $\vec{\Gamma}(G, S)$ contains a Hamiltonian circuit. Let b generate Z_m. In this case, $\mathbf{C}$ consists of the two elements

$$c_o = \langle a; 0, \ldots, 0, 0 \rangle, \text{ and}$$

$$c_1 = \langle a; 0, \ldots, 0, b \rangle$$

We now describe the graph-theoretic structure of $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$. The vertex set is $\{\langle a; b_1, \ldots, b_n \rangle : b_i \in Z_m, a \in Z_n\}$. There is an arc from $\langle a; b_1, \ldots, b_n \rangle$ to $\langle a'; b'_1, \ldots, b'_n \rangle$ (where $a, a' \in Z_n$ and $b_i \in Z_m$) when either

- $a' = a + 1$ and $b_i = b'_i$ for all i (called c_0 **arcs**) or
- $a' = a + 1$ and $b_i = b'_i$ for all $i \neq a$ and $b'_a = b_a + 1$ (called c_1 **arcs**).

Vertices that have an i in the first component are said to be in row i. The set of all vertices that have a fixed value in the base group are said to be in the same column.

Theorem 19 *The Cayley digraph $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$ has a Hamiltonian circuit for all $n, m \geq 2$.*

Proof. We first notice that $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$ consists of m copies of $\vec{\Gamma}(Z_m \wr Z_{n-1}, \mathbf{C})$ except that each column forms a cycle of length n instead of $n-1$. More formally, each of these copies is isomorphic to the quotient digraph $\vec{\Gamma}_R(Z_m \wr Z_n/\xi(Z_m \wr Z_n), \mathbf{C})$, where $\xi(G)$ denotes the center[1] of G. Each of these "altered" copies of $\vec{\Gamma}(Z_m \wr Z_{n-1}, \mathbf{C})$ is connected to a single neighboring copy via the c_1 arcs emanating from the vertices on row $n-1$. (The last copy is connected to the first copy.) This observation leads us to an inductive tool for constructing a circuit in $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$ using m copies of a Hamiltonian circuit in $\vec{\Gamma}(Z_m \wr Z_{n-1}, \mathbf{C})$. For the base case ($n = 2$) it is easily verified that[2]

$$X_2 = (c_1 c_0)^{m-1} c_1 c_1 (c_1 c_0)^{m-1} c_1 c_1$$

is a Hamiltonian circuit in $\vec{\Gamma}(Z_m \wr Z_2, \mathbf{C})$. To describe the construction of the circuit in $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$, let X_{n-1} be a circuit in $\vec{\Gamma}(Z_m \wr Z_{n-1}, \mathbf{C})$ and proceed as follows:

1. Trace the path given by X_{n-1} in $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$ except whenever we are at a vertex in row $n-1$ and X_{n-1} has not completed, we follow the c_0 arc taking us back to row 0.

2. After X_{n-1} completes, we are at a vertex on row $n-1$ and the same column at which X_{n-1} started. Continue by following the c_1 arc to the beginning of the next copy of $\vec{\Gamma}(Z_m \wr Z_{n-1}, \mathbf{C})$.

3. Repeat the first step. At the end of the traversal of the m-th copy of $\vec{\Gamma}(Z_m \wr Z_{n-1}, \mathbf{C})$ following the c_1 arc returns us to the identity element.

The above construction leads to the following recursive description of the Hamiltonian circuit for $\vec{\Gamma}(Z_m \wr Z_n, \mathbf{C})$. It is the concatenation of the n paths, $P_0, P_1, \ldots, P_{n-1}$, where

[1]The center of a group G consists of the set of all $g \in G$ such that $gg' = g'g$ for all $g' \in G$.

[2]We specify a Hamiltonian circuit by listing the appropriate sequence of generators that defines it, starting from the identity element.

P_i is defined by

$$P_i \equiv \begin{cases} (c_1\ c_0^{n-1})^{m-1}\ c_1 & \text{if } i = 0 \\ (c_1\ c_0^{n-i-1}, P_0, P_1, \ldots, P_{i-1})^{m-1}\ c_1 & \text{if } i = 1, 2, \ldots, n-1 \end{cases}$$

Notice that the circuit constructed (see Figures 8.2 and 8.3 for examples) resembles the process of base m counting through the elements of the wreath product (whence the name for the generating set). The following corollary is a straightforward extension of Theorem 19.

Corollary 6 *If G comes equipped with a generating set S such that $\vec{\Gamma}(G, S)$ has a Hamiltonian circuit then the Cayley digraph $\vec{\Gamma}(G \wr Z_n, \mathbf{C})$ has a Hamiltonian circuit.*

8.2 Iterated Wreath Products

In this section, we prove that the counting generating set for an *iterated* wreath product of cyclic groups is in fact a Hamiltonian generating set. The proof of this first requires an investigation as to when the counting generating set for $G \wr Z_n$ is minimal.

Lemma 7 *Let $S = \{s_1, s_2, \ldots, s_k\}$ be a minimal set of generators for G, and let $\mathbf{C} = \{c_0, c_1, \ldots, c_k\}$ be the corresponding set of counting generators for $G \wr Z_n$. $\mathbf{C}$ is a minimal set of generators for $G \wr Z_n$ if and only if $\vec{\Gamma}(G, S)$ does not have a circuit of length m and a circuit of length $m+1$ for any $m \geq 2$.*

Proof. For purposes of contradiction, let us suppose $\mathbf{C}$ is not a minimal set of generators for $G \wr Z_n$. Since S is minimal for G and multiplication by c_0 does not affect the elements in each component of the base group (it only permutes them), it is not possible to have a product of elements in $\mathbf{C} - \{c_i\}$ equal to c_i for $i \neq 0$. Thus, it follows that there exists a sequence $c_{k_1}, c_{k_2}, \ldots, c_{k_\ell}$ of elements in $\mathbf{C} - \{c_0\}$ for which

$$c_{k_1} \cdot c_{k_2} \cdots c_{k_\ell} = c_0.$$

By examining this product more closely, we observe the following.

- The first component in the result of the above product will equal the element $a \in Z_n$ if and only if $k_\ell = mn + 1$ for some natural number m.

- Now consider the second component of the wreath product $G \wr Z_n$ (*i.e.*, an element of the base group). Each consecutive block of n generators contributes exactly one element to the product being accrued in each component of the base group.

Thus, after m "blocks" of generators (each of length n) have been multiplied, each product that has accrued in the individual components of the base group will be of length m. The final multiplication by c_{k_ℓ} will make the length of the product in the last component (*i.e.*, the rightmost component) equal to $m + 1$. Hence, $\vec{\Gamma}(G, S)$ must have both a circuit of length m and a circuit of length $m + 1$.

Conversely, suppose that there is a set of m elements of S whose product is the identity of G: say $c_{k_1} \cdot c_{k_2} \cdots c_{k_m} = e$, and similarly, suppose that there is a set of $m + 1$ elements of S whose product is the identity of G: say $c_{i_1} \cdot c_{i_2} \cdots c_{i_m} c_{i_{m+1}} = e$. Then, we can derive c_0 as a product of the other generators of $\mathbf{C}$, as the following equation verifies:

$$c_{i_1} \cdot c_{k_1}^{n-1} \cdot c_{i_2} \cdot c_{k_2}^{n-1} \cdots c_{i_m} \cdot c_{k_m}^{n-1} \cdot c_{i_{m+1}} = c_0.$$

This establishes the non-minimality of $\mathbf{C}$. □

Theorem 20 *The iterated wreath product* $(\ldots(Z_{i_0} \wr Z_{i_1}) \wr \ldots) \wr Z_{i_n}$, *where each* $i_j > 1$, *has a Hamiltonian generating set.*

Proof. We can inductively apply Corollary 6 to obtain a Hamiltonian circuit in the Cayley digraph of this group using the counting generating set. We need only show the minimality of the generating set. Consider $(G \wr Z_{i_j}) \wr Z_{i_{j+1}}$ where $G \wr Z_{i_j}$ is a wreath product of cyclic groups equipped with the counting generating set, denoted by S. Every product of elements in S equal to the identity must have length equal to an integer multiple of i_j, for otherwise the first component of this product cannot be the identity element. Hence,

by Lemma 7, the counting generating set is minimal for $G \wr Z_{i_{j+1}}$. The result follows by induction on the number of factors. □

8.3 Extensions to Other Groups

We can define analogous "counting" generating sets for both abelian groups and semidirect products of cyclic groups (also known as *metacyclic* groups) which endow the corresponding Cayley digraphs with Hamiltonian circuits. More specifically, consider the semidirect product[3] of Z_m by Z_n, denoted $Z_n \otimes_{sd} Z_m$, with defining relations

$$a^n = e,\ b^m = e,\ a^{-1}ba = b^r \text{ where } r^n \equiv 1 \pmod{m}.$$

Every element of $Z_n \otimes_{sd} Z_m$ is uniquely specified by a pair,

$$(x, y),$$

where $x \in Z_n$ and $y \in Z_m$. Note that (a, e) generates a subgroup isomorphic to Z_n (all elements with their second component equal to the identity) and similarly, (e, b) generates a subgroup isomorphic to Z_m. The counting generating set for $Z_n \otimes_{sd} Z_m$ is defined as

$$\mathbf{C} = \{(a, e), (a, b)\},$$

and, for brevity, we let $c_0 = (a, e)$ and $c_1 = (a, b)$. Note that this generating set is minimal unless $Z_n \otimes_{sd} Z_m$ is isomorphic to a cyclic group (in this case, we would have $r = 1$ and n coprime to m). Define the following path in $\vec{\Gamma}(Z_n \otimes_{sd} Z_m, \mathbf{C})$, motivated by the definition of the circuit constructed for the wreath product with the counting generating set:

$$P = (c_1, c_0^{n-1})^m.$$

We now prove that P is in fact a Hamiltonian circuit.

[3]It is convenient for us to return to multiplicative notation for this discussion, with the identity of a group denoted by e.

$$\begin{array}{cccccc}
(e,e) & (e,b^{r^{n-1}}) & (e,b^{2r^{n-1}}) & (e,b^{3r^{n-1}}) & \cdots & (e,b^{(m-1)r^{n-1}}) \\
(a,e) & (a,b) & (a,b^2) & (a,b^3) & \cdots & (a,b^{m-1}) \\
(a^2,e) & (a^2,b^r) & (a^2,b^{2r}) & (a^2,b^{3r}) & \cdots & (a^2,b^{(m-1)r}) \\
(a^3,e) & (a^3,b^{r^2}) & (a^3,b^{2r^2}) & (a^3,b^{3r^2}) & \cdots & (a^3,b^{(m-1)r^2}) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
(a^{n-1},e) & (a^{n-1},b^{r^{n-2}}) & (a^{n-1},b^{2r^{n-2}}) & (a^{n-1},b^{3r^{n-2}}) & \cdots & (a^{n-1},b^{(m-1)r^{n-2}})
\end{array}$$

Figure 8.1: Enumeration of elements of $Z_n \otimes_{\mathrm{sd}} Z_m$, as traversed by P.

Theorem 21 *The counting generating set for $Z_n \otimes_{\mathrm{sd}} Z_m$ is a Hamiltonian generating set if and only if $Z_n \otimes_{\mathrm{sd}} Z_m$ is not isomorphic to a cyclic group.*

Proof. If $Z_n \otimes_{\mathrm{sd}} Z_m$ is isomorphic to a cyclic group, then n and m are coprime and c_1 generates the entire group. In this case, the counting generating set is not minimal and therefore cannot be a Hamiltonian generating set. On the other hand, if $Z_n \otimes_{\mathrm{sd}} Z_m$ is not isomorphic to a cyclic group then obviously $\mathbf{C}$ must be minimal. We now show that (in either case) P defines a Hamiltonian circuit in $\vec{\Gamma}(Z_n \otimes_{\mathrm{sd}} Z_m, \mathbf{C})$. Consider the enumeration of the elements of the group traversed by P as depicted in Figure 8.1.

Clearly, this is an array of group elements having n rows and m columns. Number the rows from 1 to n (proceeding from top to bottom) and number the columns from 1 to m (proceeding from left to right). The path P starts in the top left-hand corner, proceeds to the element in the second row and second column, cycles around the column to the first row, and repeats this pattern, finally returning to the starting point. We must verify that all of the elements in this table are *distinct*, proving that P is a Hamiltonian circuit. Firstly, it is clear (by examining the first components) that any two elements in different rows are distinct. We now show that any two elements from the *same* row are also distinct, giving us our result.

Let us focus on the second row. Clearly, this particular row consists of distinct elements since the order of b is m. Now observe that the second components of the elements in any other row can be obtained from the second components of the second row

by repeated conjugation by the element a. Since conjugation induces an automorphism of a group (see [55]), we conclude that the set of all elements in any given row must be distinct. □

8.4 A Topic for Further Investigation

We conclude this chapter with some remarks on the counting generating set for wreath products. It is interesting to consider whether the condition in Corollary 6 is a necessary one: Suppose S generates the group G and that $\vec{\Gamma}(G, S)$ *does not* have a Hamiltonian circuit. Is it possible that $\vec{\Gamma}(G \wr Z_n, \mathbf{C})$ has a Hamiltonian circuit? A prime candidate for investigation in regard to this problem is the group $G = (Z_2 \times Z_3) \wr Z_2$, since it is the smallest possible non-trivial counter-example. Let $Z_2 \times Z_3$ have the standard generating set consisting of the two elements $S = \{(e, a), (b, e)\}$, where a generates Z_3 and b generates Z_2. By observation (or, more formally, by the Trotter-Erdös conditions [101]), the Cayley digraph $\vec{\Gamma}(Z_2 \times Z_3, S)$ does not have a Hamiltonian circuit and it is easily verified (using Lemma 7) that the counting generating set for G is non-minimal. If we remove the redundant generator, then we have verified by exhaustive computer search that the resultant Cayley digraph of G *does not* have a Hamiltonian circuit. If, on the other hand, we include the redundant generator then the digraph becomes too large to determine its hamiltonicity by naive computer search methods. However, the resolution of this question would either disprove the converse of Corollary 6 (if the graph was found to be Hamiltonian) or provide the first example of a non-abelian group with a generating set of size at least 3 for which the Cayley digraph does not have a Hamiltonian circuit (the question of whether there exists such a group was stated as an open problem in [110]).

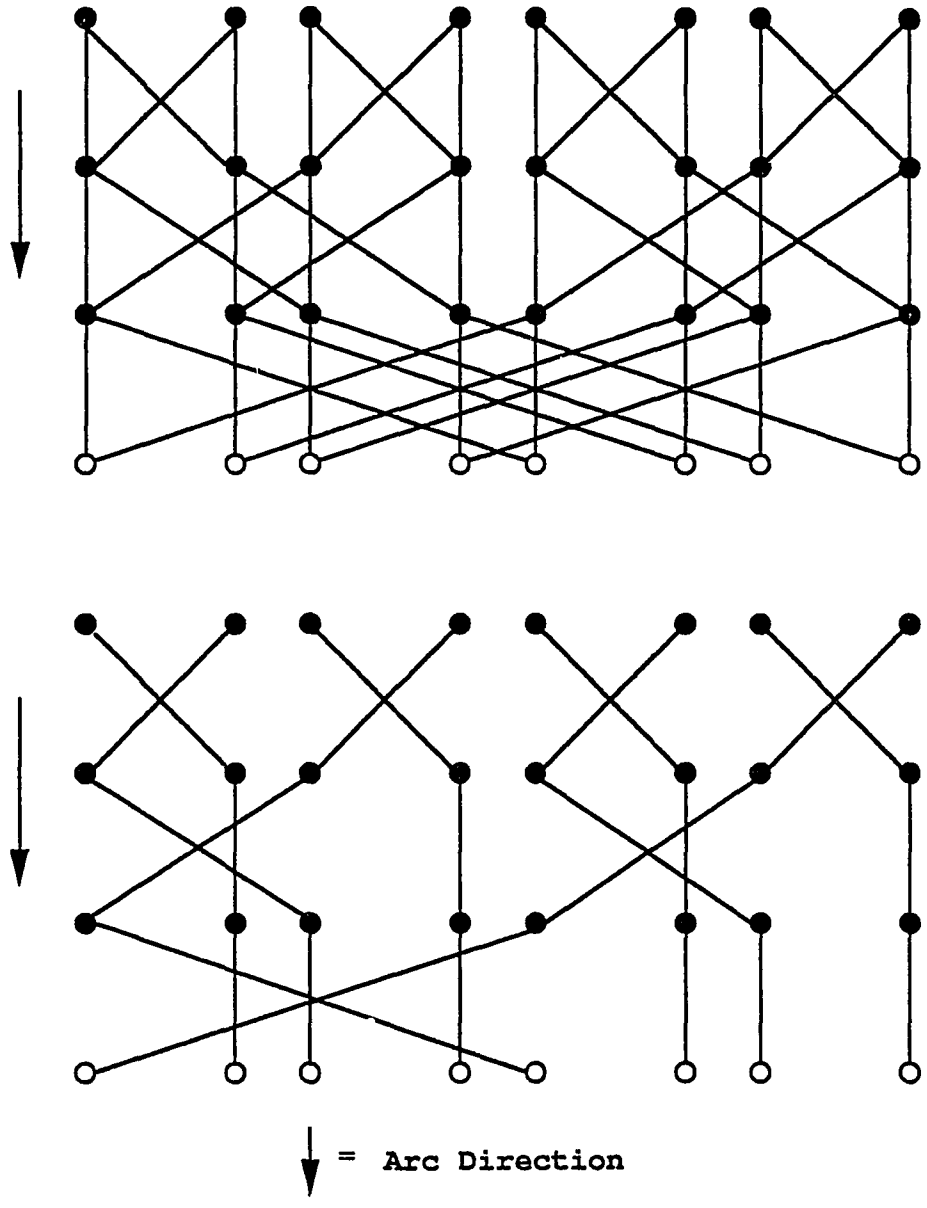


Figure 8.2: A drawing of $\Gamma(Z_2 \wr Z_3, \mathrm{C})$ and its Hamiltonian circuit

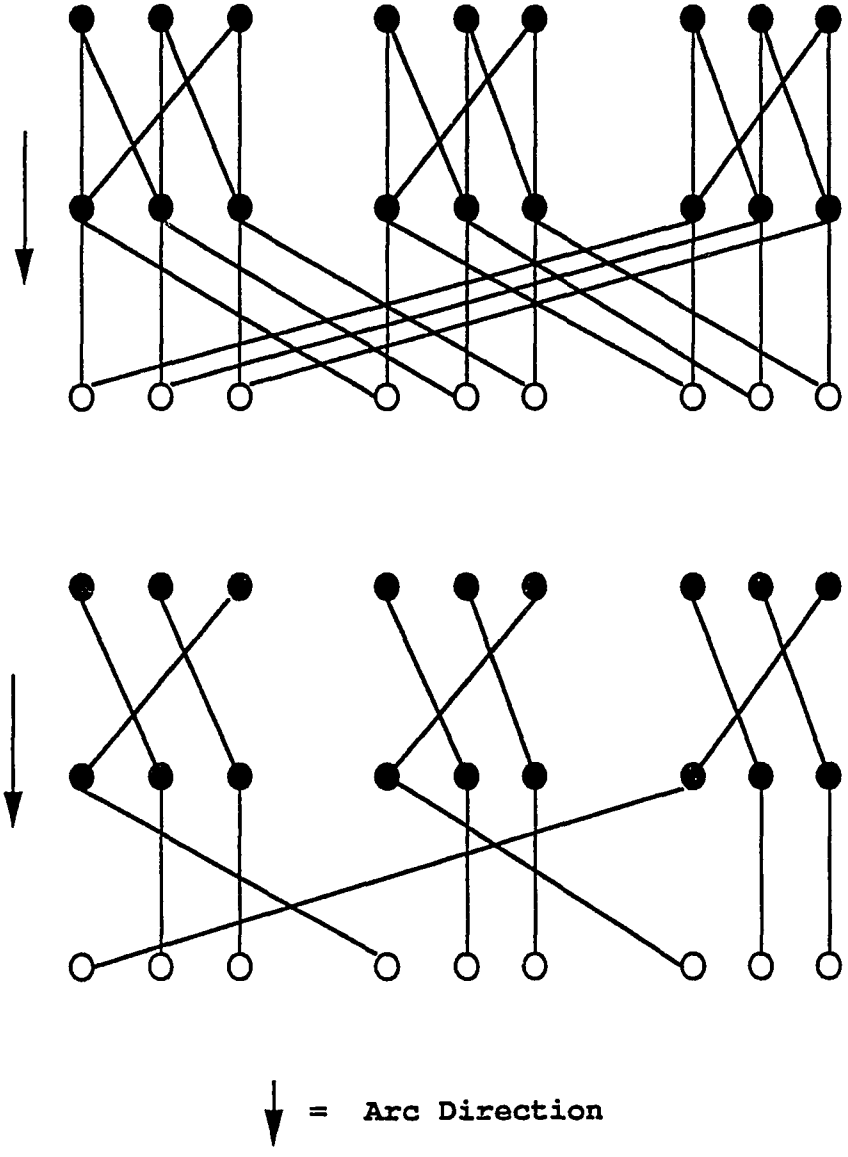


Figure 8.3: A drawing of $\Gamma(Z_3 \wr Z_2, \mathrm{C})$ and its Hamiltonian circuit

CHAPTER 9

Conclusions and Future Work

In this final chapter, we summarize the main results of the dissertation and discuss some directions for future research. The first five chapters were devoted to describing the group-theoretic approach for studying parallel networks and we demonstrated the efficacy of this approach by proving several new results pertaining to the issues of communication, reliability and inter-simulatability of Cayley networks. In particular, we discovered a general strategy for off-line permutation routing that applies to a wide class of topologies, produces efficient routes and extends to routing permutations in faulty networks. Several interesting questions arise with regard to this work. In particular, we mention the following research problems.

1. We would like to prove the following variant of the Cayley graph routing thesis: For every group G, there exists a *minimal* generating set S such that $\Gamma(G, S)$ supports efficient permutation routing (*i.e.*, there exists a congestion-free off-line routing algorithm where the length of the routes are at most a *fixed* constant of the diameter of $\Gamma(G, S)$).

2. Given a subgroup tower and a corresponding strong generating set for a group G, it is clear that the diameter of the resultant Cayley graph is bounded above by the length of the subgroup tower. When can we be sure that the diameter is bounded below by (a constant times) this quantity? This would provide better information on the strength of Theorem 9 and the results obtained by Blaha [28] on randomized routing in such Cayley graphs.

3. Recently, D. Krizanc [70] discovered a technique for improving the direct product routing algorithm for the mesh, producing routes of length at most $2.25n + 3$ and congestion 4 (at most 4 packets per processor per time step), by exploiting the fact that two independent permutations can be simultaneously routed in $3n - 3$ steps with congestion 2 (Section 3.2.2). Can this fact be further exploited to obtain an algorithm for routing a single permutation in less than $3n - 3$ steps without *any* congestion?

4. To what extent can the general strategy for finding efficient permutation routes be extended to on-line algorithms? For example, given efficient randomized routing algorithms (employing a Valiant-Brebner routing scheme [105], for example) for two networks, $\mathcal{G}$ and $\mathcal{H}$, can they be "combined" to produce an efficient randomized routing algorithm for the product network $\mathcal{G} \times \mathcal{H}$? A positive answer to this question may help to unify (and simplify) previous proofs regarding on-line routing algorithms.

In Chapter 4, we studied the connectivity of Cayley graphs and derived a general inductive tool for analyzing their connectivity. In conjunction with its corollaries, the main theorem extends previous results of Akers and Krishnamurthy [3] and Godsil [49]. In addition, the constructive nature of these results led us to suggest a new approach for studying the fault-tolerance of large-scale parallel networks by considering the connectivity of quotient graphs. Our technique is based on the fact that left coset graphs are always vertex-transitive and their connectivity is at least two-thirds that of their degree (by a theorem of Watkins [107]). An interesting avenue for future investigation is to study the *wide-diameter* of quasi-minimal Cayley networks, a new graph-theoretic parameter recently introduced by Hsu and Lyuu [64] to study fault-tolerant routing. If $\mathcal{G}$ is a graph with $\kappa(\mathcal{G}) = k$, then for each pair of vertices, u, v, consider the set of *shortest* k disjoint paths between u and v. The **wide-diameter** is defined to be the maximum length of

these shortest paths, taken over all pairs of vertices in $\mathcal{G}$. We believe that the technique developed in Chapter 4 may be applicable to the analysis of the wide-diameter of quasi-minimal Cayley graphs, using the fact that the bundles between adjacent left cosets consist of "parallel" edges that may be "spliced" into liftings of paths in the left-quotient graph.

In Chapter 5, we showed how to decompose a Cayley graph into a direct product of two smaller graphs with a small penalty in terms of dilation and congestion. We examined some special cases and discussed how the embeddings can be used to obtain efficient and uniform emulations of Cayley networks on a large range of smaller "associated" networks. One area that warrants further analysis is the general graph-theoretic question of how efficiently (in terms of dilation and congestion) an arbitrary regular graph can be decomposed into a direct product graph. This would provide a valuable measurement of the strength of our embedding results for Cayley graphs.

Chapter 6 presented a one-to-one embedding of the order-n deBruijn network into the order-n hypercube network with dilation $\lfloor 2/5n \rfloor$ and congestion 2. This is the best analytical result known for this problem, although the question of whether there exists a dilation $o(\log N)$ embedding remains unresolved. Another pressing problem is to find a lower bound for embedding a deBruijn graph into a hypercube. It is widely believed that the lower bound is, in fact, within a constant factor of the *diameter* of the hypercube. However, it is not even known whether the lower bound is greater than 2. (It is clear that the deBruijn graph cannot be a subgraph of the hypercube since the hypercube is bipartite and the deBruijn graph is not bipartite.) We suspect that the solution to this problem will require some radical new ideas in the area of graph embeddings.

In the latter half of Chapter 6, we suggested a generalization of the technique implicit in our embedding of the deBruijn network into the hypercube. This technique subsumes a previously discovered embedding of the butterfly network in the deBruijn network ([10])

and provides a new embedding for the mesh network into the deBruijn network. We suspect that there may be other fruitful applications, such as embedding higher-order deBruijn networks into hypercubes.

In Chapter 7, we designed and analyzed a reconfiguration algorithm for faulty hypercube networks. This algorithm is the first deterministic solution for this problem, and is based on a greedy search procedure (in conjunction with backtracking) for finding an embedding of a fault-free hypercube in a faulty one. An open question is whether this scheme can be successfully applied to other networks, such as star networks and pancake networks.

In the final chapter, we addressed the question of Hamiltonian circuits for Cayley digraphs of iterated wreath products, verifying a conjecture of Klerlein for this class of groups. An interesting related topic for future research is the search for Hamiltonian circuits in quasi-minimal Cayley graphs. In particular, we would like to determine conditions under which a Hamiltonian circuit in a left quotient graph can be "lifted" to a Hamiltonian circuit in the corresponding Cayley graph. (An analogous study was undertaken by Alspach [7] for a different type of "quotient" graph.) For example, consider the case of an order-n butterfly graph, whose left coset graph is an order-n hypercube. The existence of a Hamiltonian circuit in the butterfly that passes though each coset (*i.e.*, column) exactly once reduces to finding a Hamiltonian circuit in the hypercube with the property that each consecutive edge is in a consecutively numbered dimension modulo n. Preliminary computer searches indicate that such a "gray code" for the hypercube does exist, but we have not yet found a general proof.

Finally, we mention that there are many other fertile problems in the theory of parallel networks that may be approached from a group-theoretic angle. For example, how does symmetry affect the ability to obtain a VLSI layout of a parallel network? More generally, does the presence of symmetry in a graph make it easier to analyze its properties? We

also suggest that tools from combinatorial group theory may provide new machinery to attack the mapping problem for parallel algorithms. Many classes of algorithms can be specified by specific classes of directed acyclic graphs [81]. If we restrict ourselves to trees (as in the case of divide-and-conquer algorithms), then by regarding a tree as a subgraph of a Cayley graph of an appropriate *free* group it seems possible to use the fact that finite groups are quotients of free groups in order to *uniformly* map any instance of our algorithm (which, over all of its instances may be an infinite tree) onto a fixed finite physical Cayley network. This avenue of investigation seems to hold much promise for future research efforts.

BIBLIOGRAPHY

[1] G.B. Adams III and H.J. Siegel (1982): The extra-stage cube: A fault-tolerant interconnection network for supersystems. *IEEE Transactions on Computers*, Vol. 31, No. 5, 443-454.

[2] B. Aiello, T. Leighton, B. Maggs and M. Newman (1990): Fast algorithms for bit-serial routing on a hypercube. *2nd Symposium on Parallel Algorithms and Architectures*, 55-64.

[3] S.B. Akers and B. Krishnamurthy (1987): On group graphs and their fault tolerance. *IEEE Transactions on Computers*, Vol. 36, 885-888.

[4] S.B. Akers and B. Krishnamurthy (1989): A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, Vol. 38, No. 4, 555-566.

[5] G.S. Almasi and A. Gottlieb (1989): Highly Parallel Computing. Benjamin/Cummings, Redwood City, CA.

[6] N. Alon and V.D. Milman (1985): λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38, 73-88.

[7] B. Alspach (1989): Lifting Hamilton cycles of quotient graphs. *Discrete Math.*, 78, 25-36.

[8] F. Annexstein (1989): Fault tolerance of hypercube-derivative networks. *1st Symposium on Parallel Algorithms and Architectures*, 179-188.

[9] F. Annexstein and M. Baumslag (1989): Limitations on constructing expanders with Cayley graphs. Submitted for publication.

[10] F. Annexstein, M. Baumslag and A.L. Rosenberg (1990): Group action graphs and parallel architectures. *Siam Journal on Computing*, Vol. 19, No. 3, 544-569.

[11] L. Babai, G. Hetyei, W.M. Kantor, A. Lubotzky and A. Seress (1990): On the diameter of finite groups. *31st Symposium on Foundations of Computer Science*, 857-865.

[12] L. Babai, W.M. Kantor and A. Lubotzky (1988): Small diameter Cayley graphs for finite simple groups. Typescript.

[13] R.G. Babb II (1988): Programming Parallel Processors. Addison-Wesley, Reading, MA.

[14] E. Bannai and T. Ito (1973): On finite Moore graphs. *Journal of Fac. Sci. Univ. Tokyo*, 20, 191-208.

[15] M. Baumslag (1991): On the fault-tolerance of quasi-minimal Cayley networks. *Advances in Computing and Information - ICCI '91*, LNCS 497, 431-442.

[16] M. Baumslag and F. Annexstein (1991): A unified framework for off-line routing in parallel networks. To appear in *Mathematical Systems Theory.* A preliminary version appeared in *2nd Symposium on Parallel Algorithms and Architectures*, 398-406.

[17] M. Baumslag, M.C. Heydemann, J. Opatrny and D. Sotteau (1991): Embeddings of shuffle-like graphs in hypercubes. *Parle '91.*

[18] M. Baumslag and A.L. Rosenberg (1991): Processor-time tradeoffs for Cayley graph interconnection networks. *Proceedings of the 2nd Distributed Memory Computing Conference*, Portland, Oregon.

[19] BBN Laboratories Inc. (1986): Butterfly parallel processor overview. *BBN Report No. 6148.*

[20] B. Becker and H.U. Simon (1986): How robust is the n-cube? *27th Symposium on Foundations of Computer Science.*

[21] V.E. Beneš (1965): Mathematical Theory of Connecting Networks. *Mathematics in Science and Engineering*, Volume 17, Academic Press, New York.

[22] F. Berman (1983): Parallel programming with limited resources. *Proceedings of the Conference on Information Sciences and Systems*, The Johns Hopkins University, 675-679.

[23] J.-C. Bermond, F. Comellas and D.F. Hsu (1991): Distributed loop computer networks: A survey. To appear in *Journal of Parallel and Distributed Computing.*

[24] J.-C. Bermond and T. Tzvieli (1991): Minimal diameter double-loop networks: dense optimal families. To appear in *Networks.*

[25] J.-C. Bermond, C. Delorme and J.-J. Quisquater (1986): Strategies for interconnection networks: Some methods from graph theory. *Journal of Parallel and Distributed Computing* 3, 433-449.

[26] S.N. Bhatt, F.R.K Chung, J.-W. Hong, F.T. Leighton and A.L. Rosenberg (1988): Optimal simulations by butterfly networks. *20th Symposium on Theory of Computing.*

[27] S.N. Bhatt, F.R.K. Chung, F.T. Leighton and A.L. Rosenberg (1986): Optimal simulations of tree machines. *27th Symposium on Foundations of Computer Science*, 274-282.

[28] K. Blaha (1989): Algorithms for Permutation Groups and Cayley Networks. Doctoral dissertation, University of Oregon, Eugene.

[29] H. L. Bodlaender (1988): The complexity of finding uniform emulations on paths and ring networks. *Information and Computation*, 86, 87-106.

[30] H.L Bodlaender and J. van Leeuwen (1986): Simulations of large networks on smaller networks. *Inform. Comput.* 71, 143-180.

[31] B. Bollobas (1978): Extremal Graph Theory. Academic Press, New York.

[32] J. Bruck, R. Cypher and D. Soroker (1990): Running algorithms efficiently on faulty hypercubes. *2nd Symposium on Parallel Algorithms and Architectures*, 37-44.

[33] L. Campbell, G.E. Carlsson, V.Faber, M.R. Fellows, M.A. Langston, J.W. Moore, A.P. Mullhaupt and H.B. Sexton (1988): Dense symmetric networks from linear groups and codes. Computer Science Technical Report CS-88-192, Washington State University.

[34] G.E. Carlsson, J.E. Cruthirds, H.B. Sexton and C.G. Wright (1985): Interconnection networks based on a generalization of the cube-connected cycles. *IEEE Transactions on Computers*, Vol. 34, No.8, 769-772.

[35] R.M. Chamberlain (1988): Gray codes, fast fourier transforms and hypercubes. *Parallel Computing* 6, 225-233.

[36] M.Y. Chan (1989): Embedding of d-dimensional grids into optimal hypercubes. *1st Symposium on Parallel Algorithms and Architectures*, 52-57.

[37] M.Y Chan and S.J. Lee (1990): Fault-tolerant permutation routing in hypercubes. University of Texas (Dallas) Technical Report, UTDCS-4-90.

[38] F.R.K. Chung (1986): Diameters of communication networks. In *Mathematics of Information Processing*, M. Anshel and W. Gewritz (eds.). Proceedings of Symposia Applied Mathematics, Vol. 34, American Mathematical Society, Providence, Rhode Island, 1-14. *Proceedings of Applied Mathematics.* Vol. 34.

[39] C. Clos (1959): A study of non-blocking switching networks. *Bell System Technical Journal* 32, 406-424.

[40] H.S.M. Coxeter and W.O.J. Moser (1972): Generators and Relations for Discrete Groups, 3rd edition. Springer-Verlag, Berlin.

[41] W.J. Daly (1987): Performance analysis of k-ary n-cube interconnection networks. Typescript, Massachussetts Institute of Technology.

[42] W.J. Daly and C.L. Seitz (1987): Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36, 547-553.

[43] D. Eppstein and Z. Galil (1988): Parallel algorithmic techniques for combinatorial computation. *Ann. Rev. Comput. Sci.*, 3, 233-283.

[44] V. Faber (1990): Global communication algorithms for hypercubes and other Cayley coset graphs. To appear in *Siam Journal on Discrete Math.*.

[45] M.R. Fellows (1985): Encoding graphs in graphs. Doctoral dissertation, University of California, San Diego.

[46] J.P. Fishburn and R.A. Finkel (1982): Quotient networks. *IEEE Transactions on Computers*, Vol. 31, 288-295.

[47] M. Furst, M. Hopcroft and E.M. Luks (1980): Polynomial-time algorithms for permutation groups. *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, Vol. 21, 36-41.

[48] W.H. Gates and C.H. Papadimitriou (1979): Bounds for sorting by prefix reversal. *Siam Journal on Discrete Math.*, No. 27, 47-57.

[49] C.D. Godsil (1981): Connectivity of minimal Cayley graphs. *Arch. Math.*, Vol. 37, 473-476.

[50] D. Gorenstein (1968): Finite Groups. Harper & Row, New York.

[51] A.C. Green (1975): Structure of vertex-transitive graphs. *Journal of Combinatorial Theory, Series B*, 18, 1-11.

[52] D. Greenberg (1990): Private Communication.

[53] D. Greenberg and S. Bhatt (1990): Routing multiple paths in hypercubes. *2nd Symposium on Parallel Algorithms and Architectures*, 45-54.

[54] D.S. Greenberg, L.S. Heath and A.L. Rosenberg (1990): Optimal embeddings of butterfly-like graphs in the hypercube. To appear in *Mathematical Systems Theory.*

[55] M, Hall, Jr. (1959): The Theory of Groups. Macmillan, New York.

[56] Y.O. Hamidoune (1984): On the connectivity of Cayley digraphs. *European Journal of Combinatorics* 5, 309-312.

[57] J. Hastad, F. Leighton and M. Newman (1987): Fast computation using faulty hypercubes. *19th Symposium on the Theory of Computation.*

[58] J. Hastad, F. Leighton and M. Newman (1989): Fast computation using faulty hypercubes. *21st Symposium on the Theory of Computation.*

[59] M. Herbordt, C. Weems and J. Corbett (1990): Message-passing algorithms for a SIMD torus with coteries. *2nd Symposium on Parallel Algorithms and Architectures*, 11-21.

[60] W.D. Hillis (1986): The Connection Machine. The MIT Press, Cambridge, MA.

[61] D.F. Hsu and J. Shapiro (1991): Bounds for the minimal number of transmission delays in double loop networks. To appear in *Journal of Combinatorics, Information and System Sciences.*

[62] D.F. Hsu and J. Shapiro (1991): A census of tight one-optimal double loop networks. To appear in *Graph Theory, Combinatorics, Algorithms and Applications.*

[63] D.F. Hsu and X-D. Jia (1991): Extremal problems in the construction of distributed loop networks. To appear in *Siam Journal on Discrete Math.*

[64] D.F. Hsu and Y-D. Lyuu (1991): A graph-theoretical study of transmmision delay and fault-tolerance. Preprint.

[65] M. Imase and Y. Manabe (1988): Fault-tolerant routings in a κ-connected network. *Information Processing Letters*, 27, 171-175.

[66] W. Imrich (1979): On the connectivity of Cayley graphs. *Journal of Combinatorial Theory Series B*, 26, 323-326.

[67] M. Jerrum (1983): The complexity of finding a minimal-length generating sequence. Technical Report CRS-139-83, Univ. of Edinburgh.

[68] J.B. Klerlein (1978): Hamiltonian cycles in Cayley color graphs, *Journal of Graph Theory*, 2, 65-68.

[69] R. Koch, F.T. Leighton, B. Maggs, S. Rao and A.L. Rosenberg (1989): Work-preserving emulations of fixed-connection networks. *21st Symposium on Theory of Computing.*

[70] D. Krizanc (1991): A note on off-line routing on a mesh-connected processor array. *Advances in Computing and Information - ICCI '91*, LNCS 497, 418-420.

[71] D. Krizanc, S. Rajasekaran and T. Tsantilas (1988): Optimal routing algorithms for mesh-connected processor arrays. In *VLSI Algorithms and Architectures*, Proceedings of the 3rd AWOC 88, LNCS 319, J.H. Reif (editor), 411-422.

[72] M. Kunde (1988): Routing and sorting on mesh-connected arrays. In *VLSI Algorithms and Architectures*, Proceedings of the 3rd AWOC 88, LNCS 319, J.H. Reif (editor), 423-433.

[73] M. Livingston, Q. Stout, N. Graham and F. Harary (1987): Subcube fault-tolerance in hypercubes. Technical Report CRL-TR-12-87, University of Michigan Computing Research Laboratory.

[74] A. Lubotzky, R.Philips and P. Sarnak (1988): Ramanujan graphs. *Combinatorica*, 8, 261-277.

[75] W. Mader (1971): Eine Eigenschaft der Atome endlicher Graphen. *Arch. Math. (Basel)* 22, 333-336.

[76] G.A. Margulis (1988): Explicit group-theoretic constructions of combinatorial schemes and their applications for the construction of expanders and concentrators. *Probl. of Info. Transm.*, 24/1, 39-46.

[77] K. Mulmuley, U.V. Vazarani and V.V. Vazarani (1987): Matching is as easy as matrix inversion. *19th Symposium on Theory of Computing*, 345-354.

[78] D. Nassimi and S. Sahni (1980): An optimal routing algorithm for mesh-connected parallel computers. *Journal of the Association for Computing Machinery*, Vol. 27, No. 1, 6-29.

[79] D. Nassimi and S. Sahni (1982): Parallel algorithms to set up the Beneš permutation network. *IEEE Transactions on Computers*, Vol. 31, No. 2, 148-154.

[80] C.H. Papadimitriou and K. Steiglitz (1982): Combinatorial Optimization. Prentice-Hall, New Jersey.

[81] C.H. Papadimitriou and M. Yannakakis (1988): Towards an architecture-independent analysis of parallel algorithms. *Symposium on Theory of Computation*, 510-513.

[82] D.S. Parker (1980): Notes on shuffle/exchange-type switching networks. *IEEE Transactions on Computers*, Vol. 29, No. 3, 213-222.

[83] D.S. Parker and C.S. Raghavendra (1984): Tha Gamma network. *IEEE Transactions on Computers*, Vol. 33, No. 4, 367-373.

[84] D. Peleg and J. Ullman (1989): An optimal synchronizer for the hypercube. *Siam Journal on Computing.*, Vol. 18, No. 4, 740-747 .

[85] D.K. Pradhan and M.R. Samatham (1989): The deBruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI. Doctoral dissertation, University of Massachusetts (Amherst).

[86] F.P. Preparata and J.E. Vuillemin (1981): The cube-connected cycles: a versatile graph for parallel computation. *Communications of the ACM*, 24, 300-309.

[87] A. Ranade (1987): How to emulate shared memory. *28th Symposium on Foundations of Computer Science*, 496-504.

[88] F. Roberts (1984): Applied Combinatorics. Prentice Hall, New Jersey.

[89] D.J.S. Robinson (1980): A Course in the Theory of Groups. Springer-Verlag, Berlin.

[90] A.L. Rosenberg (1971): Data graphs and addressing schemes. *Journal of Computer and System Sciences* 5, 193-238.

[91] A.L. Rosenberg (1972): Addressable data graphs. *JACM*, 19, 309-340.

[92] A.L. Rosenberg (1983): The Diogenes approach to fault-tolerant arrays of processors. *IEEE Transactions on Computers*, Vol. 32, 902-910.

[93] A.L. Rosenberg (1988): The product-shuffle network: toward reconciling shuffles and butterflies. To appear in *Journal of Discrete Applied Mathematics*.

[94] Y. Saad and M.H. Schultz (1989): Data communication in hypercubes. *Journal of Parallel and Distributed Computing* 6, 115-135.

[95] G. Sabidussi (1964): Vertex-transitive graphs. *Monatsh. Math.* 63, 124-127.

[96] N. Santoro and R. Khatib (1985): Labelling and implicit routing in networks. *The Computer Journal*, Vol. 28, No. 1, 5-8.

[97] E. J. Schwabe (1990): On the computational equivalence of hypercube-derived networks. *2nd Symposium on Parallel Algorithms and Architectures*, 388-397.

[98] D. Slepian (1952): Two theorems on a particular crossbar switching network. Unpublished manuscript.

[99] M.A. Sridhar (1988): On the connectivity of deBruijn graphs. *Information Processing Letters* 27, 315-318.

[100] R. Stong (1987) : On Hamiltonian cycles in Cayley graphs of wreath products. *Discrete Math.* 65, 75-80.

[101] W.T. Trotter, Jr. and P. Erdös (1978): When the cartesian product of directed cycles is Hamiltonian. *Journal of Graph Theory* 2, 137-142.

[102] D. Tzvieli (1988): Minimal diameter double-ring networks I: Some very large infinite optimal families. Typescript, Louisiana State University.

[103] J. D. Ullman (1983): Computational Aspects of VLSI. Computer Science Press, Rockville, MD.

[104] L.G. Valiant (1982): A scheme for fast parallel computation. *Siam Journal on Computing*, 11, 2, 350-361.

[105] L.G. Valiant and G.J. Brebner (1981): Universal schemes for parallel computation. *Proceedings of the Symposium on Theory of Computation*, 263-277.

[106] M.E. Watkins (1969): Some classes of hypoconnected vertex-transitive graphs. *Recent Progress in Combinatorics; Proceedings of the Third Waterloo Conference on Combinatorics*, W.T. Tutte & C.St.J.A Nash-Williams, editors, Academic Press, New York.

[107] M.E. Watkins (1970): Connectivity of transitive graphs. *Journal of Combinatorial Theory*, 8, 23-29.

[108] M.E. Watkins (1985): Computing the connectivity of circulant graphs. *Congressus Numerantium*, 49, 247-258.

[109] P.M. Winkler (1987): The metric structure of graphs. *Surveys in Combinatorics*, C. Whitehead, editor, London Math. Society Lecture Notes Series 123, 197-221.

[110] D. Witte and D. Gallian (1984): A survey: Hamiltonian cycles in Cayley graphs. *Discrete Math.*, 51, 293-304.

[111] C.L. Wu and T.Y. Feng (1980): On a class of multistage interconnection networks. *IEEE Transactions on Computers*, Vol. 29, No. 8, 694-702.

[112] H.P. Yap (1986): Some Topics in Graph Theory. London Math. Society Lecture Note Series 108, Cambridge University Press.

[113] M. Yoeli (1962): Binary ring sequences. *Amer. Math. Monthly 69*, 852-855.

[114] H.J. Zassenhaus (1958): The Theory of Groups, 2nd edition. Chelsea Publishing Company, New York.