ALGORITHMS FOR PERMUTATION GROUPS AND CAYLEY NETWORKS

by

KENNETH D. BLAHA

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

August 1989

APPROVED: Eugene M. Luks

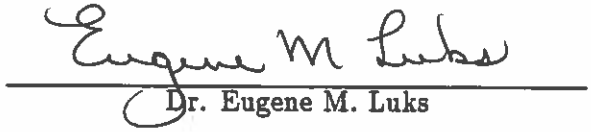
Dr. Eugene M. Luks

An Abstract of the Dissertation of

Kenneth D. Blaha for the degree of Doctor of Philosophy

in the Department of Computer and Information Science to be taken August 1989

Title: ALGORITHMS FOR PERMUTATION GROUPS AND CAYLEY NETWORKS

Approved: 

Dr. Eugene M. Luks

Bases, subgroup towers and strong generating sets (SGSs) have played a key role in the development of algorithms for permutation groups. We analyze the computational complexity of several problems involving bases and SGSs, and we use subgroup towers and SGSs to construct dense networks with practical routing schemes.

Given generators for $G \leq Sym(n)$, we prove that the problem of computing a minimum base for G is NP-hard. In fact, the problem is NP-hard for cyclic groups and elementary abelian groups. However for abelian groups with orbits of size less than 8, a polynomial time algorithm is presented for computing minimum bases.

For arbitrary permutation groups a greedy algorithm for approximating minimum bases is investigated. We prove that if $G \leq Sym(n)$ with a minimum base of size k, then the greedy algorithm produces a base of size $\Omega(k \log\log n)$.

We show that the decision analogs to the following two algebraic problems are LOGSPACE-complete for P: computing a greedy base for $G \leq Sym(n)$, and "sifting" $\sigma \in G$ through an arbitrary SGS for G. In contrast, we prove that for any polynomial

subgroup tower of a solvable group one can find an SGS for which sifting is in NC.

Because of their inherent symmetry Cayley graphs are an attractive model for parallel processing networks. Unfortunately, Jerrum has shown that computing a minimal route for an arbitrary Cayley graph is PSPACE-complete.

We use subgroup towers and SGSs to construct Cayley networks with "failsoft" routing algorithms, and we adapt Valiant's permutation routing algorithm to run on the directed Cayley networks. Normal towers are used to define Cayley graphs and routing algorithms that perform well, as long no more than $d-1$ processors fail (d the degree of the graph). For several Cayley network families we exhibit a universal broadcast algorithm that runs in optimal time.

These same techniques are used to analyze nonsymmetric networks. In particular, we prove that Leland and Solomon's moebius graph is isomorphic to a quotient Cayley graph. This information is used to efficiently compute minimum routes and determine the diameter of the moebius graph.

VITA

NAME OF THE AUTHOR: Kenneth D. Blaha

PLACE OF BIRTH: New Prague, Minnesota

DATE OF BIRTH: October 5, 1954

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon
University of Minnesota at Morris

DEGREES AWARDED:

Doctor of Philosophy, 1989, University of Oregon
Master of Science, 1984, University of Oregon
Master of Science, 1981, University of Oregon
Bachelor of Arts, 1978, University of Minnesota at Morris

AREAS OF SPECIAL INTEREST:

Algebraic Algorithms, Cayley Graphs and Multiprocessor Networks

PROFESSIONAL EXPERIENCE:

Graduate Researcher, Department of Computer and Information Science, University of Oregon, 1988-89

Graduate Research Assistant, Department of Computer and Information Science, University of Oregon, 1985-88

Graduate Teaching Fellow, Department of Computer and Information Science, University of Oregon, 1983-85

AWARDS AND HONORS:

Tektronix Fellowship Award, 1988-89

PUBLICATIONS:

BLAHA, K. Minimum bases for permutation groups: The greedy approximation. Tech. Rep. CIS-TR-86-16, University of Oregon, 1986.

BLAHA, K. Finding a minimum base for permutation groups is NP-hard. In *Congressus Numerantium* (1987), vol. 58, pp. 141–150.

ACKNOWLEDGEMENTS

First and foremost I wish to express my sincere gratitude to my adviser, Eugene M. Luks. His tutelage, insight and intuition have been invaluable. I also wish to thank the other members of my committee, William M. Kantor, Andrzej Proskurowski and Chris B. Wilson. Special thanks to Bill for all the advice he has given me over the years, and to László Babai for his contribution to the proof of Proposition 2.6.

I cannot offer adequate words of thanks to my wife, Barbara, for her hours with this manuscript. I am indebted to Dave Meyer for his help in preparing this document. Finally, I wish to thank the faculty, staff and colleagues in the CIS department who have made my stay here at the University a pleasant experience.

DEDICATION

To my family

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER I

INTRODUCTION

Motivation and Overview

Over the last thirty years there has been considerable effort focused on the development of algorithms for permutation groups. Since the size of a permutation group G on n points can be exponential in n, it is usually necessary to specify the group by a set of generators. Fortunately, a generating set of size $O(n)$ exists for every permutation group $G \leq Sym(n)$. Given such a succinct representation for G the question arises as to whether we can find efficient solutions to basic queries about the group. Using a base and strong generating set (SGS) Sims devised a method of storing the group that satisfies the following properties:

(a) it uses no more than a poly(n) (polynomial in n) space

(b) given any $g \in Sym(n)$ we can determine in poly(n) time if $g \in G$

(c) we can run through the elements of G one at a time without repetitions

The base is used to define a subgroup tower $G = G^0 \geq G^1 \geq G^2 \geq \cdots \geq G^k = \{id\}$, and the subgroup tower is used to define an SGS [41]. Once an SGS for the group is known membership can be determined easily by "sifting" through the coset representatives (a complete description of the sifting process is given in the next section). Although Sims described a practical algorithm for computing an SGS, it was not until 1980 [20] that a

version of Sims' algorithm was proved to run in polynomial time $O(n^6)$.

Shortly after that, Knuth suggested a clever implementation of Sims' algorithm that he analyzed to run in $O(n^5 \log\log n)$ time [26]. Using Babai's result [3], which gives a linear limit on the length of subgroup chains in $Sym(n)$, this bound was later improved to $O(n^5)$. Subsequently, Jerrum described an algorithm for computing a base and SGS with running time $O(n^5)$ [25]. Jerrum's algorithm also reduced the number of strong generators from $O(n^2)$ to $O(n)$.

A careful analysis of Jerrum's algorithm shows that the running time is $O(n^3k^2)$ [6], where k is the size of the base produced by the algorithm. One can show that the same bound holds for Knuth's algorithm.

The size of the base not only affects the running time of algorithms that use a base and SGS, but in many cases determines the space needed to work within the group. This follows from the observation that every permutation in G is uniquely determined by its action on a base B. By focusing our attention on B, we can save a considerable amount of space and time when computing bases and SGSs for subgroups of G.

These savings are particularly important when the base is significantly smaller than the degree. Cannon points out that excluding the alternating group, permutation representations of simple groups have this property [9]. For example, Held's group (order 4,030,387,200) has a permutation representation of degree 2058, but the group has a base of size 8. Large group theory packages, such as CAYLEY, rely on small bases to manipulate such groups efficiently.

The size of a base is not a group invariant and, as we shall see, may vary by as much as a $\log n$ factor. Thus, the question arises as to whether we can find efficient algorithms

to compute "small" bases. Cameron considered a greedy algorithm for computing small bases as early as 1982 [8]. The Greedy approach is simply to pick a point b_i that will force the subgroup G^i to be as small as possible. Since $|G^i| = \frac{|G^{i-1}|}{r}$, where r is the size of the G^{i-1}-orbit of b_i, we see that the Greedy algorithm picks a point b_i from a largest G^{i-1}-orbit.

This Greedy algorithm was independently discovered and implemented in a paper on symmetry and backtracking [7]. At that time Finkelstein posed the question as to whether or not the Greedy algorithm always computed a minimum bases [19]. In Chapter II we prove that the minimum base problem is NP-hard. The construction in the proof of this result is used to build groups for which the Greedy algorithm fails to find a minimum base. We also show that under certain restrictions the minimum base problem is solvable in polynomial time, but not by the Greedy algorithm.

Under the assumption that P$\neq$NP we know that there is no polynomial time solution to the minimum base problem. It is natural to ask if the Greedy algorithm is a good approximation heuristic. In Chapter III we answer this question by comparing the size of a Greedy base to the size of a largest nonredundant base, and to the size of a minimum base. We also compare the Greedy algorithm to another heuristic that has been suggested for finding small bases.

So far our discussion has focused on sequential algorithms for permutation groups. McKenzie and Cook were among the first to study parallel algorithms for permutation groups [37]. They showed that for abelian groups, the fundamental problem of group membership (Is $g \in G$?) is in NC. They also conjectured that the membership problem is P-complete (complete for P with respect to logspace reductions) for arbitrary groups,

and hence not likely to be in NC.

This conjecture was based on the following observations. First, the sifting process used in membership testing seems inherently sequential. Second, the tower of subgroups generated by a base may have length linear in n. It was later shown through a series of papers, [35], [34] and [4], that membership is, in fact, in NC. However, the tower of subgroups used to perform the sifting was quite different from the tower produced by a base.

As a consequence of this work it was shown that a base and SGS could be constructed in NC, but the parallel sifting problem remained open [4]. In Chapter IV we address this problem. We show that there exists a base and SGS for which the sifting problem is P-hard. In contrast to this result, we show that for solvable groups and polynomial subgroup towers one can always find an SGS for which sifting is in NC. In this chapter we also answer the following question, " Is there a parallel algorithm for computing Greedy bases?" We show that it is highly unlikely that such an algorithm exists by proving that the deterministic Greedy base problem is P-complete.

In the second part of this dissertation we show how towers of subgroups and SGSs can be used to study parallel processing networks. The interconnection topology of a multiprocessor network is usually modeled as a graph where vertices represent processing elements and edges correspond to communication lines. Three (static) parameters that are often considered when evaluating these networks are: the diameter k, the degree d and the number of vertices $N(d,k)$ with respect to a fixed value of d and k. We say that a graph is dense if $N(d,k)$ is large. The upper bound for $N(d,k)$ is called the Moore bound and is given by the formula $d+d(d-1)+d(d-1)^2+\cdots+d(d-1)^{k-1}$. It is known that the

Moore bound is not obtainable, except for a few small values of k and d [5]. The degree is an important parameter since it is a measure of the amount of hardware needed by the architecture. The diameter is important since message delay is proportional to it; and we would like the overhead incurred by message delay small in comparison to the advantage gained from parallel computation.

Other factors such as routing, congestion, regularity, fault tolerance and symmetry, to mention a few, are also important. Networks like the Chordal Ring, presented by Arden and Lee [2] in 1981, and the Cosmic Cube, produced by the Intel Corporation, realized the importance of these other factors. Both of these networks are not only moderately dense, but are also accompanied by a "failsoft" routing algorithm. A routing algorithm is failsoft if it is possible to route around a faulty processor (vertex) with little or no degradation in message delay. The Moebius graph [30], developed in 1982 by Leland and Solomon, is a degree three network which also has a routing algorithm and is more dense then the Chordal Ring or the Cosmic Cube.

Most of the networks suggested in the last ten years (e.g., [1], [46], [15], [16], [30], [29], [38], [39], and [40]) share some common characteristics. They are all regular, symmetric, and are families of graphs with an underlying structure. In 1985 Carlsson, Cruthirds, Sexton and Wright observed that both the Cube and the Cube-connected cycles (CCC) could be viewed as Cayley graphs [10] and [12]. A Cayley graph $\Gamma = (G, W)$ is a graph where the vertex set G is a finite group. The edge set is determined by the set of generates W for the group. Two vertices $g, h \in G$ are adjacent if and only if $gw = h$, where w is an element of the generating set W. Note that the graph is directed, regular and symmetric.

Since then it has been noted that many of the families of regular graphs mentioned above may be viewed as Cayley graphs [11]. Thus there was a common thread that linked these networks together. Carlsson, Cruthirds, Sexton and Wright used the algebraic structure of the Cayley graph to generalize the CCC and produce new graphs that were more dense then any known to date. Not only were these graphs dense, but since they were Cayley graphs, they were symmetric and the authors believed they would have the same desirable properties that the other networks possessed. Although the authors believed that a good routing algorithm for these graphs should exist, none was presented. The question is, how does one take advantage of the underlying group structure to find good routing algorithms?

The problem of finding a minimum route on a Cayley graph is related to the problem of finding a minimal length generating sequence (MLGS) (smallest word in the generators) of an arbitrary group element. Goldreich and Even showed that MLGS is NP-hard [17]. Since then Jerrum has shown that it is PSPACE-complete [24]. Consequently, it would be unreasonable to attempt to solve the routing problem for an arbitrary group and an arbitrary set of generators.

Thus not only is the choice of the group important when constructing Cayley graphs, but the choice of generators is a critical consideration. In Chapter V we show how SGSs can be used to construct Cayley graphs with failsoft routing algorithms, and how Valiant's permutation routing algorithm [44] can be adapted to run on the directed Cayley networks. We also show how normal towers can be used to define Cayley graphs and routing algorithms that perform well, as long as no more than $d-1$ processors fail. We conclude this section with several examples of Cayley networks. In one of the examples the underlying

group is a Sylow-2 subgroup of the symmetric group on n elements (n a power of two). In this case the generators are chosen with great care so that sifting could be applied. Also, techniques from Jerrum [25] are used to reduce the size of the generating set without significantly increasing the diameter of the Cayley graph.

In Chapter VI we extend Faber's work on universal broadcast schemes [18]. In particular, we show how certain Cayley networks constructed from SGSs can perform universal broadcast in optimal time. We obtain these optimal results with Cayley graphs constructed from abelian groups and Sylow-2 subgroups. We also answer the question stated by Faber as to whether every quotient Cayley graph can perform universal broadcasts in optimal time.

We denote a quotient Cayley graph (QCG) by $\Gamma(G, H, W)$. The graph has vertex set equal to the right cosets of H in G, and two vertices Hg, Hh are connected if and only if $Hh = Hgw$ for some $w \in W$. Intuitively the QCG, $\Gamma(G, H, W)$ is constructed from the Cayley graph $\Gamma(G, W)$ by merging all vertices that lie in the same coset into one vertex.

A number of the networks previously mentioned, such as the Moebius and Shuffle-exchange, are not symmetric. Hence, these graphs cannot be viewed as Cayley graphs. It has been pointed out that some of these graphs can be realized as QCGs. In Chapter VII we show that a network is isomorphic to a QCG if and only if it satisfies a certain "labeling" property (defined later). We use this result to show that the Moebius graph can be realized as a QCG.

Using this isomorphism we answer two open problems concerning the Moebius graph. First, we give an efficient algorithm for computing optimal routes on the Moebius graph. Second, we use the optimal routing algorithm to find the diameter of the graph. To prove

that our routing algorithm is optimal we map the routing problem on the Moebius graph to an equivalent problem on a larger Cayley graph. The problem is then solved on the Cayley graph and mapped back to the Moebius graph.

Definitions and Background

We assume a basic familiarity with what is commonly called "complexity theory", such as can be found in [23] and [21]. Throughout this dissertation we write "log" for $\log_2$ and "ln" for $\log_e$. The symbols Z and Z_m denote the integers and integers modulo m, respectively.

The group of all permutations of an n-element set Ω is denoted $\mathrm{Sym}(\Omega)$, or $\mathrm{Sym}(n)$ if the specific set is irrelevant. We write $|G|$ for the order of G and $H \leq G$ if H is a subgroup of G. The index, $|G:H|$, of H in G is the integer equal to $|G|/|H|$.

If $H \leq G$, then we define an equivalence relation on G in which two elements $g, h \in G$ are equivalent if and only if $gh^{-1} \in H$. The equivalence classes of G under this relation are called the cosets of H in G. We say that g is a coset representative of the (right) coset $Hg = \{hg | h \in H\}$. A set $U \subseteq G$ of size $|G:H|$ is a complete set of coset representatives of H in G if every element in G is equivalent to a (unique) element in the set. We say that H is normal in G, $H \trianglelefteq G$, if $Hg = gH$ for all $g \in G$.

By the degree of $G \neq \{id\}$ we mean the number of points moved by G. Let $\omega \in \Omega$, then we call $\{\omega^g | g \in \mathrm{G}\}$ the G-orbit of ω. We say that $\Delta \subseteq \Omega$ is fixed by G if $\Delta^g = \Delta$ for all $g \in G$. Each $g \in G$ induces a permutation on Δ, which we denote by g^Δ. We call the totality of the g^Δ's formed for all $g \in G$ the constituent, G^Δ, of G on Δ.

For $A \subseteq \Omega$, we define the set $G_A = \{g \in G | \forall a \in A,\ a^g = a\}$, called the

point-wise set stabilizer of A. If A consists of a single point, a, then we write $G_A = G_a$.

For any abstract group H, we define the homomorphism $\mathcal{R} : H \rightarrow \mathrm{Sym}(H)$ such that $\mathcal{R}(h)$ acts on H via right multiplication. That is, for each "point" $x \in H$, $x^{\mathcal{R}(h)} = xh$. We call $\mathcal{R}(H)$ the right regular representation of H. For additional background information on permutation groups see either [36], or [47].

The following definitions are due to Sims and may be found in [42]. A base for G is a sequence of points $B = b_1, b_2, \ldots, b_k$, $b_i \in \Omega$, such that the only element in G fixing all of the b_i is the identity. We say that base B has size k, and we denote the size of a smallest base for G by $\mathcal{M}(G)$. The tower of subgroups

$$G = G^0 \geq G^1 \geq \cdots \geq G^k = \{id\},$$

where $G^i = G_{\{b_1,\ldots,b_i\}}$, $1 \leq i \leq k$ is called the chain of stabilizers of G relative to B. We call a base nonredundant if each of the inclusions $G^{i-1} \geq G^i$ is proper.

This tower has three characteristics that are essential for computational purposes. First, the tower has length no more than $n - 1$. Second, $|G^{i-1} : G^i|$ is polynomial in n for $1 \leq i \leq k$ (infact, $|G^{i-1} : G^i| \leq n - i + 1$). Any subgroup tower satisfying this second condition is called polynomial. Third, given $g \in G$ we can determine in linear time if $g \in G^i$, $1 \leq i \leq k$.

It is these three properties that allow us to compute an SGS for G relative to B in polynomial time. An SGS is a subset S of G such that G^{i-1} is generated by $S \cap G^{i-1}$, $1 \leq i \leq k$. An SGS of particular interest to us is comprised of coset representatives, U_i of G^i in G^{i-1} such that $id \in U_i$ for $1 \leq i \leq k$. Unless otherwise stated we shall

always assume that an SGS is of the form $U = \bigcup_{i=1}^{k} U_i$. This particular SGS is of interest because every element $g \in G$ can be written uniquely as $g = u_k u_{k-1} \cdots u_1$, where $u_i \in U_i$. Moreover, this factorization can be realized in $O(k^2)$ time by "sifting" g through the coset representatives.

The first step in the sifting process is to find a $u_1 \in U_1$ such that gu_1^{-1} fixes 1 (i.e., $gu_1^{-1} \in G^1$). Next we find $u_2 \in U_2$ such that $gu_1^{-1}u_2^{-2}$ fixes 2. Continuing in this manner we find $g = u_k u_{k-1} \cdots u_1$ where $u_i \in U_i$. If we store the inverses of the SGS and use the appropriate data structure we can find $u_i \in U_i$ in $O(k)$ time. Thus the time needed to sift g is $O(k^2)$. This sifting technique will be used to perform routing on the Cayley networks.

We should point out that the membership problem is solved by sifting. Suppose that we are given $\sigma \in Sym(n)$ and we wish to know, " Is $\sigma \in G$?" Then we sift σ through the SGS for G relative to B. If at some stage of the sifting process we cannot find a $u_i \in U_i$ such that $gu_1^{-1}u_2^{-1} \cdots u_i^{-1} \in G^i$, then we know that $\sigma \notin G$. Observe that we must check that $gu_1^{-1}u_2^{-1} \cdots u_k^{-1} = id$; thus the time needed to solve the membership problem, given the SGS, is $O(nk)$.

Mathematical Tools

The following 3 facts are used in the paper; proofs of these statements may be found in either [36] or [47].

Fact 1.1 Let $G \leq Sym(\Omega)$ and $\omega \in \Omega$. If r is the size of the G-orbit of ω, then $|G : G_\omega| = r$.

Fact 1.2 Let G be a finite group and $g \in G$. Then $\mathcal{R}(G)$ is isomorphic to G ($\mathcal{R}(G) \cong G$) and $\mathcal{R}(G)_g = \{id\}$.

Fact 1.3 If $G \leq Sym(\Omega)$, and $B = b_1, b_2, \ldots, b_k$ is a base for G, then (by Lagrange's Theorem) $|G| = |G^0 : G^1||G^1 : G^2| \cdots |G^{k-1} : G^k|$.

Lemma 1.4 Given $G = \langle \sigma \rangle \leq \mathrm{Sym}(n)$ and base $B = b_1, b_2, \ldots, b_k$, define r_i to be the size of the G-orbit (cycle of σ) containing b_i. Then $G^m = \langle \sigma^r \rangle$ where $r = lcm\{r_1, r_2, \ldots, r_m\}$, $1 \leq m \leq k$.

Proof: $G^m = \langle \sigma^j \rangle \Leftrightarrow j$ is the smallest positive integer such that σ^j fixes $b_1, \ldots, b_m$. But σ^j fixes $b_i \Leftrightarrow r_i$ divides j. □

We shall use the following construction in Chapters II, III and IV. Let X be a fixed finite set, and $\{\sigma_x | x \in X\}$ a fixed set of generators for the group $(Z_2)^{|X|}$. For $Y \subseteq X$ define $G(Y) = \langle \sigma_x | x \in Y \rangle$.

The right regular action $\mathcal{R} : G(Y) \rightarrow \mathrm{Sym}(G(Y))$ is extended to an action $\mathcal{R}_Y : G(X) \rightarrow \mathrm{Sym}(G(Y))$ in which $\{\sigma_x | x \in X \setminus Y\}$ act trivially. Suppose now that $\mathcal{C}$ is a collection of subsets of X. Let $\Omega_{\mathcal{C}} = \dot{\bigcup}_{Y \in \mathcal{C}} G(Y)$ (the disjoint union of the sets $G(Y)$, $Y \in \mathcal{C}$). Then the $\mathcal{R}_Y$ for $Y \in \mathcal{C}$, induce an action $\mathcal{R}_{\mathcal{C}} : G(X) \rightarrow \mathrm{Sym}(\Omega_{\mathcal{C}})$, where $\omega^{\mathcal{R}_{\mathcal{C}}(\sigma)} = \omega^{\mathcal{R}_Y(\sigma)}$ if $\omega \in G(Y)$. Note that, if $X = \bigcup_{Y \in \mathcal{C}} Y$, then $\mathcal{R}_{\mathcal{C}}$ is faithful. The reader may wish to examine the example given at the end of this chapter.

Now, given $Z \subseteq X$ we let $G_{\mathcal{C}}(Z)$ denote the permutation group $\mathcal{R}_{\mathcal{C}}(G(Z))$. We use the next two lemmas to analyze bases of $G_{\mathcal{C}}(Z)$.

Lemma 1.5 Let $Z \subseteq X$, $Y \in \mathcal{C}$ and $\omega \in G(Y)$, then $G_{\mathcal{C}}(Z)_\omega = G_{\mathcal{C}}(Z \setminus Y)$.

Proof: $G_{\mathcal{C}}(Z)_\omega = \mathcal{R}(\{\sigma \in G(Z) | \omega^{\mathcal{R}_{\mathcal{C}}(\sigma)} = \omega\} = \mathcal{R}(\langle \sigma_x | x \in Z \setminus Y \rangle)$.

The first equality follows from the definition of point stabilizer and the second from Fact 1.2. □

Lemma 1.6 Let $W = Y \cap Z$, where $Z \subseteq X$, and $Y \in C$. Then the set $G(Y)$ has $|G(Y) : G(W)|$ $G_C(Z)$-orbits each of size $|G(W)|$.

Proof: Let $H \leq L$, and $R : H \rightarrow \mathrm{Sym}(L)$ be a homomorphism such that H acts on L via right multiplication (i.e., $l^{R(h)} = lh$). Then the $R(H)$-orbits are the left cosets (lH) of H in L. To finish the proof note that $G(W) \leq G(Y)$, and that the action of $G_C(Z)$ on $G(Y)$ is exactly the action of $R(G(W))$ on $G(Y)$. □

The Greedy Algorithm

We find a Greedy base for a permutation group G by repeatedly picking b_i from a largest orbit of G^{i-1}. Since $|G^i| = \frac{|G^{i-1}|}{r}$, where r is the size of the G^{i-1}-orbit of b_i, we see that the Greedy heuristic selects a point b_i that forces $|G^i|$ to be as small as possible.

A naive implementation of the Greedy heuristic using Knuth's algorithm would result in a running time of $O(k'k^2n^3)$, where k' is the size of the Greedy base, n the degree, and k the size of the base produced by Knuth's algorithm. In [7, pages 17 and 19], it was observed that Jerrum's algorithm could be modified to include the Greedy heuristic without increasing the asymptotic running time of the algorithm.

Below we outline Jerrum's algorithm [25] for computing a base and strong generating set, and explain how the algorithm can be modified to include the Greedy heuristic. The input to Jerrum's algorithm is a group $G \leq \mathrm{Sym}(\{1, 2, \ldots, n\})$ specified by generators. The output of the algorithm is the base $B = 1, 2, \ldots, n$ and a data structure that contains a strong generating set for G.

Jerrum's Algorithm:

(1) For $i = 1$ to n do

(1.1) Using generators for G^{i-1} compute a set of coset representatives for G^i in G^{i-1}

(1.2) Update the data structure

(1.3) Compute a set of $O(n^2)$ Schreier generators for G^i

(1.4) Reduce the Schreier generators to a set of $O(n)$ generators for G^i

We need only a slight modification to include the Greedy heuristic in Jerrum's algorithm.

Jerrum's Algorithm with the Greedy Heuristic:

(1) $i = 1$

(2) While $G^{i-1} \neq id$ do begin

(2.1) Pick a point b_i from a largest G^{i-1}-orbit

(2.2) Using generators for G^{i-1} compute a set of coset representatives for G^i in G^{i-1}

(2.3) Update the data structure

(2.4) Compute a set of $O(n^2)$ Schreier generators for G^i

(2.5) Reduce the Schreier generators to a set of $O(n)$ generators for G^i

(2.6) $i = i + 1$

The running time of this algorithm is dominated by step (2.5). Thus, Jerrum's algo-

rithm may be modified to include the Greedy heuristic without increasing the asymptotic running time of the algorithm.

We conclude this section with an example that serves two functions. First, it gives the reader a concrete example of how the set X and the collection $\mathcal{C}$ of subsets of X are used to construct the permutation group $G_{\mathcal{C}}(X)$. Second, it shows that the Greedy algorithm fails to find a minimum base for the group $G_{\mathcal{C}}(X)$.

Example 1.1 Let $X = \{a,b,c,d,e,f\}$, $Y_1 = \{a,b\}$, $Y_2 = \{c,d\}$, $Y_3 = \{e,f\}$, and $Y_4 = \{a,c,e\}$. Let $\mathcal{C} = \{Y_1, Y_2, Y_3, Y_4\}$. Following the construction outlined above we define the following elementary abelian 2-groups: $G(X) = \langle \sigma_x | x \in X \rangle$, $G(Y_1) = \langle \sigma_a, \sigma_b \rangle$, $G(Y_2) = \langle \sigma_c, \sigma_d \rangle$, $G(Y_3) = \langle \sigma_e, \sigma_f \rangle$, $G(Y_4) = \langle \sigma_a, \sigma_c, \sigma_e \rangle$.

Recall that $\Omega_{\mathcal{C}}$ is the disjoint union of the $G(Y_i)$ $i = 1,2,3,4$, and

$$G_{\mathcal{C}}(X) \leq Sym(G(Y_1)) \times Sym(G(Y_2)) \times Sym(G(Y_3)) \times Sym(G(Y_4)).$$

The monomorphism $\mathcal{R}_{\mathcal{C}} : G(X) \rightarrow Sym(\Omega_{\mathcal{C}})$ maps the generators of $G(X)$ to generators of $G_{\mathcal{C}}(X)$: $\mathcal{R}_{\mathcal{C}}(\sigma_a) = (\sigma_a, id, id, \sigma_a)$, $\mathcal{R}_{\mathcal{C}}(\sigma_b) = (\sigma_b, id, id, id)$, $\mathcal{R}_{\mathcal{C}}(\sigma_c) = (id, \sigma_c, id, \sigma_c)$, $\mathcal{R}_{\mathcal{C}}(\sigma_d) = (id, \sigma_d, id, id)$, $\mathcal{R}_{\mathcal{C}}(\sigma_e) = (id, id, \sigma_e, \sigma_e)$, $\mathcal{R}_{\mathcal{C}}(\sigma_f) = (id, id, \sigma_f, id)$.

Let $G = G_{\mathcal{C}}(X)$, then G is a permutation group of degree 20, and $|G| = 64$. Let $B = b_1, b_2, b_3, b_4$ be a sequence of points from $\Omega_{\mathcal{C}}$, such that $b_i \in G(Y_i)$, $1 \leq i \leq 4$. Let $G = G^0 \geq G^1 \geq G^2 \geq G^3 \geq G^4$ be the chain of stabilizers of G relative to B. Using Remark 1.5 we display, in Table 1, generators for each subgroup in the chain.

Since $G^4 = \{id\}$ we know that B is a base for G. Furthermore, since we know the generators for each group in the chain, we can use Fact 1.2 and Remark 1.6 to exhibit, in

Table 1: Generators for the Subgroup Chain

Fixed Points	Group Generators
none	$G^0 = \mathcal{R}_C(\langle \sigma_a, \sigma_b, \sigma_c, \sigma_d, \sigma_e, \sigma_f \rangle)$
b_1	$G^1 = \mathcal{R}_C(\langle \sigma_b, \sigma_d, \sigma_f \rangle)$
$b_1 b_2$	$G^2 = \mathcal{R}_C(\langle \sigma_d, \sigma_f \rangle)$
$b_1 b_2 b_3$	$G^3 = \mathcal{R}_C(\langle \sigma_f \rangle)$
$b_1 b_2 b_3 b_4$	$G^4 = \{id\}$

Table 2, the orbit structure of each group in the chain.

Table 2: Orbit Structure

Group	Orbit Structure			
G^0	$G(Y_1)$	$G(Y_2)$	$G(Y_3)$	$G(Y_4)$
G^1	$\{1, \sigma_b\}\{\sigma_a, \sigma_a\sigma_b\}$	$\{1, \sigma_d\}\{\sigma_c, \sigma_c\sigma_d\}$	$\{1, \sigma_f\}\{\sigma_e, \sigma_e\sigma_f\}$	trivial
G^2	trivial	$\{1, \sigma_d\}\{\sigma_c, \sigma_c\sigma_d\}$	$\{1, \sigma_f\}\{\sigma_e, \sigma_e\sigma_f\}$	trivial
G^3	trivial	trivial	$\{1, \sigma_f\}\{\sigma_e, \sigma_e\sigma_f\}$	trivial
G^4	trivial	trivial	trivial	trivial

Any base for G produced by the Greedy Algorithm must begin with a point $b_1 \in G(Y_4)$, since $G(Y_4)$ is the largest G-orbit. The reader may verify that the base B is one that the Greedy Algorithm could produce. In fact, it is not hard to see that the Greedy Algorithm will always produce a base for G of size 4. A minimum base for G has size 3 and consists of one point from each of the 3 sets $G(Y_1)$, $G(Y_2)$ and $G(Y_3)$.

CHAPTER II

MINIMUM BASES FOR PERMUTATION GROUPS

We demonstrated, in Example 1.1, that the Greedy algorithm fails to find a minimum base. In the first section of this chapter we prove that the minimum base problem is, in fact, NP-hard. The corresponding decision problem of determining whether there exists a base of size at most N (for a given positive integer N) is NP-complete. Moreover, the problem remains NP-complete even if we restrict G to be either a cyclic group or an elementary abelian group with orbits size no more than 8.

We prove, in the next section, that for abelian groups this bound on the size of the orbits is sharp. That is, if G is an abelian group with orbits of size less than 8, then we can find a minimum base for G in polynomial time. Our algorithm uses Lovász's result [31], in which a maximum matching of a linear 2-polymatroid is found, in polynomial time, to handle abelian semisimple groups with orbits of size 4 and 6.

Finding Minimum Bases is NP-hard

We prove that the minimum base problem is NP-hard by showing that the corresponding decision problem is NP-complete. The decision problem small base (SB) is defined as follows:

SB Input: $G \leq \mathrm{Sym}(n)$ given by generators and a positive integer $N \leq n$.

Question: Does there exist a base for G of size no more than N?

We show that even when the group G is restricted to cyclic groups or elementary abelian groups, the SB problem is NP-complete. It is not difficult to show that the SB problem is in NP. Guess a base for G, $B = b_1, b_2, \ldots, b_k$, and check that $k \leq N$. Then use your favorite algorithm (Sims, Knuth, or Jerrum) to verify that B is a base for G.

To show that SB is complete for NP we describe a polynomial time reduction of exact cover by 3-sets (X3C) to SB [21]. We denote an instance of X3C by (Y, M), where Y is a finite set of order $3q$ and M is a collection of 3-element subsets of Y. The question is, "Does M contain a subcollection M' such that every element of Y is contained in exactly one member of M'?".

Theorem 2.1 SB is NP-complete even if G is constrained to be a cyclic group.

Proof: It suffices to show that X3C reduces to SB, where the group constructed for the SB problem is a cyclic group. Let (Y, M) be an instance of X3C with $|Y| = 3q$ and $|M| = r$. We may assume, without loss of generality, that each $y \in Y$ is contained in at least one $m \in M$. Let $P = \{p_1, p_2, \ldots, p_{3q}\}$ be the set of the first $3q$ primes. Define $f : Y \rightarrow P$ such that f is injective. For each 3-set $m = \{x, y, z\} \in M$ let $s_m = f(x)f(y)f(z)$.

Let $n = \Sigma_{m \in M} s_m$, and construct $\sigma \in \mathrm{Sym}(n)$ with cycle decomposition consisting of r disjoint s_m-cycles C_m, $m \in M$. Then we create an instance of SB, with $G = \langle \sigma \rangle$ and $N = q$. This is a polynomial time reduction, since the prime number theorem implies n is $O(r(q \log q)^3)$.

Suppose $B = b_1, b_2, \ldots, b_k$ is a base for G, where b_i is a point in cycle C_{m_i}, and $k \leq q$. Let $s = lcm\{s_{m_1}, s_{m_2}, \ldots, s_{m_k}\}$, then by Lemma 1.4 B is a base for G if and only if $s = p_1 p_2 \cdots p_{3q}$ ($|G| = p_1 p_2 \cdots p_{3q}$). Since each s_m is a product of exactly 3 primes it follows that $s = |G|$ if and only if $k = q$ and the s_{m_i} are all relatively prime, $1 \leq i \leq k$. However, the s_{m_i} are relatively prime if and only if the sets $m_1, m_2, \ldots, m_k$ are disjoint. Thus, $B = b_1, b_2, \ldots, b_k$ is a base for G with $k \leq q$ if and only if $k = q$ and 3-sets $m_1, m_2, \ldots, m_k$ cover Y. □

We can apply the construction in the NP-completeness proof to build cyclic groups for which the Greedy Algorithm fails to find a minimum base. In fact, we can use the failure of the "greedy approach" for Exact Two Cover to generate groups for which the Greedy Algorithm always fails. Exact Two Cover (also known as the complete matching problem for graphs) is solvable in polynomial time using non-greedy methods e.g., [27].

Example 2.2 Let (Y, M) be an instance of Exact Two Cover where $Y = \{a, b, c, d\}$ and M contains the 2-sets: $\{b, d\}, \{b, c\}$ and $\{a, d\}$. If we mimic the construction in the proof of Theorem 2.1 and map a, b, c, d to the primes 2,3,5,7 respectively, then the permutation σ will have cycle decomposition consisting of 3 disjoint cycles of sizes 21, 15 and 14.

Using Lemma 1.4 one checks that a minimum base for $G = \langle \sigma \rangle$ is comprised of one point from the cycle of size 15 and one point from the cycle of size 14. The Greedy algorithm starts by fixing a point b in the G-orbit of size 21. The group $G_b = \langle \sigma^{21} \rangle$ has 10 nontrivial orbits, 3 of size 5 and 7 of size 2. The Greedy algorithm selects two more points; first, a point in an orbit of size 5 is fixed, and then a point in an orbit of size 2. Thus, the Greedy algorithm will always produces a base of size 3.

Of course it is quite easy to construct cyclic groups of smaller order and degree for which the Greedy algorithm fails. In fact, we can find examples that involve only two primes.

Example 2.3 Let G be the cyclic group generated by

$$\sigma = (1,2,\ldots,8)(9,10,\ldots,17)(18,19,\ldots,29).$$

The Greedy algorithm first fixes a point, say $b = 18$, in the G-orbit of size 12. By Lemma 1.4 $G_b = \langle \sigma^{12} \rangle$. Now G_b has 4 orbits of size 2 and 3 orbits of size 3. Next the Greedy algorithm fixes a point in a 3-orbit, and then a point in an orbit of size 2 resulting in a Greedy base of size 3. A minimum base for G has size 2 (e.g., $B = 1,9$).

In the above reduction of X3C to SB the size of the orbits of the cyclic group increased, as the problem size of X3C increased. One might wonder if it is possible to solve the SB problem efficiently for groups that are restricted to have bounded orbits. Theorem 2.2 suggests that this is not the case.

Theorem 2.2 SB is NP-complete even if G is constrained to be an elementary abelian 2-group with orbits of size 8.

Proof: Let (Y, M) be an instance of X3C with $|Y| = 3q$. We assume, without loss of generality, that each $y \in Y$ is contained in at least one $m \in M$. Now we use the notation outlined in section 2.1 to define the group $G_M(Y) \leq \mathrm{Sym}(\Omega_M)$. Recall that $G_M(Y)$ is generated by the set $\{\mathcal{R}(\sigma_y) | y \in Y\}$, and $\Omega_M = \dot{\bigcup}_{m \in M} G(m)$.

Let $G = \langle \mathcal{R}(\sigma_y) | y \in Y \rangle$, and $N = q$ be our instance of SB. This is a polynomial time reduction, since $|\Omega_M| = 8|M|$ and there are only $|Y|$ generators. Let $B = b_1, b_2, \ldots, b_k$ ($k \leq q$) be a sequence of points with $b_i \in G(m_i)$. Define sets $Y_0 = Y$, and $Y_i = (Y_{i-1} \setminus m_i)$ for $1 \leq i \leq k$. Then by Lemma 1.5 we have $G^i = G_M(Y_i)$ for $1 \leq i \leq k$.

By definition, B is a base for G if and only if $G^k = 1$, and $G^k = 1$ if and only if $Y_k = \emptyset$. Since each $m \in M$ has cardinality 3, it follows that $Y_k = \emptyset$ if and only if $k = q$, and the m_i are disjoint, $1 \leq i \leq k$. Thus, B is a base for G with $k \leq q$ if and only if $k = q$ and $m_1, m_2, \ldots, m_k$ cover Y. □

Remark 2.3 We can use an analogous proof to show that the statement of Theorem 2.2 holds if we replace 2 with any fixed prime p, and 8 with p^3.

Finding Minimum Bases for Abelian Groups with Small Orbits

In the preceding section we saw that the problem of finding a minimum base remains NP-hard even if we restrict ourselves to elementary abelian 2-groups with orbits of size no more than 8. We now show that for abelian groups this bound on the size of the orbits is sharp. That is, if G is an abelian permutation group with orbits of size less than 8, then we can find a minimum base for G in polynomial time. For convenience let us call this problem AMB_7.

The algorithm is divided into four principal stages. In each stage we focus our attention on a subgroup $H \leq G$ that satisfies the following two properties. First, we can compute a minimum base for H in polynomial time. Second, any minimum base B for H can be extended to a minimum for G. We then replace G with H_B and begin the next phase of the algorithm. The following is an outline of the algorithm. The input to the

algorithm is an instance G of AMB_7 specified by generators.

Algorithm AMB_7:

(1) Find a minimum base B_1 for the subgroup of G that
fixes all the points in orbits of size 5 and 7

(2) Find a minimum base B_2 for the Frattini subgroup of G_{B_1}

(3) Find a minimum base B_3 for $(G_{B_1B_2})^{\Delta}$, where Δ
is the union of all $G_{B_1B_2}$-orbits of size 4 and 6

(4) Find a minimum base for $G_{B_1B_2B_3}$

The following proposition describes how step (1) of the algorithm is accomplished, and proves that the base we find can be extended to a minimum base for G.

Proposition 2.4 In polynomial time we may reduce any instance of AMB_7 to the problem of finding a minimum base for an abelian permutation group G', where all of the G'-orbits have size 2,3,4 or 6.

Proof: Let $G = \langle \Phi \rangle$ be an instance of AMB_7. Let Δ_1 be the union of all the G-orbits of size 7, let Δ_2 be the union of all the G-orbits of size 5, and let Δ_3 be the union of the remaining G-orbits. Then by the fundamental theorem of abelian groups, $G = G^{\Delta_1} \times G^{\Delta_2} \times G^{\Delta_3}$. By raising the generators of G to the appropriate power we find generators for the groups G^{Δ_i}, $i = 1,2,3$.

$$G^{\Delta_1} = \langle \phi^{30} | \phi \in \Phi \rangle$$

$$G^{\Delta_2} = \langle \phi^{42} | \phi \in \Phi \rangle$$

$$G^{\Delta_3} = \langle \phi^{35} | \phi \in \Phi \rangle$$

Observe that B is a minimum base for G if and only if $B \cap \Delta_i$ is a minimum base for G^{Δ_i} $1 \leq i \leq 3$. Thus, to compute a minimum base for G it will suffice to compute a minimum base for each G^{Δ_i}, $i = 1, 2, 3$.

The groups G^{Δ_1} and G^{Δ_2} are elementary abelian p-groups (p a prime) with orbits of size p. By Facts 1.1 and 1.3 any nonredundant base for these groups is a minimum base. Hence we may use the Greedy algorithm to compute a minimum base for G^{Δ_1} and G^{Δ_2}. Note that we can compute a minimum base for G^{Δ_1} and G^{Δ_2} without computing generators for the two subgroups. To find the desired base it suffices to run the Greedy algorithm on the generators for G and focus our "attention" only on the points in the set $\Delta_1 \cup \Delta_2$. □

The following proposition is helpful in verifying the correctness of steps (2) and (3) of the algorithm AMB_7. Loosely speaking, the proposition shows that we can ignore orbits of prime size.

Proposition 2.5 In polynomial time we may reduce any instance of AMB_7 to the problem of finding a minimum base for an abelian permutation group G', where all of the G'-orbits have size 4 or 6.

Proof: Let $G = \langle \Phi \rangle$ be an instance of AMB_7. By Proposition 2.4 we may assume, without loss of generality, that all of the G-orbits have size 2,3,4 or 6. Let Δ_1 be the union of all the G-orbits of size 4 and 6. Let Δ_2 be the union of all the G-orbits of size 2 and 3.

Let $|G| = 2^s 3^t$ and let $G = G_0 \geq G_1 \geq \cdots \geq G_k = \{1\}$ be the chain of stabilizers of G relative to base $B = b_1, b_2, \ldots, b_k$. If we define $I \subset \{1, 2, \ldots, k\}$ such that $[G_{i-1} : G_i]$ is

nonprime if and only if $i \in I$, then $k = s + t - |I|$. To minimize k we must maximize the size of the set I. It follows from Fact 1.1 that $|I|$ is completely determined by the group G^{Δ_1}. Hence, any minimum base for G^{Δ_1} can be extended (by the Greedy algorithm) to a minimum base for G. □

Proposition 2.6 In polynomial time we may reduce any instance of AMB_7 to the problem of finding a minimum base for a semisimple abelian permutation group G', where all of the G'-orbits have size 4 or 6.

Proof: Let $G = \langle \Phi \rangle$ be an instance of AMB_7. By Proposition 2.4 and Proposition 2.5 we may assume, without loss of generality, that all of the G-orbits have size 4 or 6. If O is a G-orbit, then the constituent G^O is isomorphic to either $Z_2 \times Z_2$, Z_4 or Z_6. Let Δ_1 be the union of all the G-orbits of size 4 such that $G^O \simeq Z_4$, and let Δ_2 be the union of all the remaining G-orbits. By the fundamental theorem of abelian groups we know that there exists integers r, s and t such that $G \simeq (Z_4)^r \times (Z_2)^s \times (Z_3)^t$.

The function $F : G \rightarrow G$ defined by $F(h) = h^6$ is a homomorphism since G is abelian. The group $F(G)$ is called the Frattini subgroup of G. Note that $F(G) \leq G^{\Delta_1}$, and $F(G) \simeq (Z_2)^r$.

Any minimum base for G must contain r points from Δ_1 that constitute a base for $F(G)$. The first step of the reduction is to find r points $B = b_1, b_2, \ldots b_r$ such that B is a base for $F(G)$. This can be accomplished by running the Greedy algorithm on $F(G)$, or by running the Greedy algorithm on the group G^{Δ_1} and selecting the first r points that are fixed by the algorithm. The base B that we obtain for $F(G)$ is by no means unique. To finish the proof we must prove that any minimum base for $F(G)$ can be extended to a

minimum base for G.

It will suffice to show that if B and B' are two minimum bases for $F(G)$, then $\mathcal{M}(G_B) = \mathcal{M}(G_{B'})$. The groups G_B and $G_{B'}$ are abelian semisimple (since they contain no elements of order 4), and both groups have order 2^s3^t. Hence, each group is isomorphic to $(Z_2)^s \times (Z_3)^t$. If we could prove that $G_B = G_{B'}$ we would be done. Unfortunately this statement is not true. Consider, for example, the group $\langle (1,2,3,4)(5,6,7,8)\ ,\ (1,2,3,4)(5,8,7,6) \rangle$ and let $B = 1$ and $B' = 5$. Instead we prove a weaker statement that is sufficient to imply that $\mathcal{M}(G_B) = \mathcal{M}(G_{B'})$.

By the proof of Proposition 2.5 we know that $\mathcal{M}(G_B)$ is completely determined by the action of the group on the G_B-orbits of size 4 and 6. All of the G_B-orbits in Δ_1 have size 2 or less. Thus $\mathcal{M}(G_B)$ is determined by the action of G_B on Δ_2. A similar argument holds for $G_{B'}$. To finish the proof we need only show that $G_B^{\Delta_2} = G_{B'}^{\Delta_2}$

Note that both $F(G) \cap G_B$ and $F(G) \cap G_{B'}$ are trivial. Thus, we may conclude that both $K = F(G) \times G_B$ and $K' = F(G) \times G_{B'}$ are subgroups of G. Both groups are elementary abelian and $|K| = |K'| = 2^{r+s}3^t$. Thus it follows that $K = ker(F) = K'$. Finally $F(G) \leq G^{\Delta_1}$ implies that $G_B^{\Delta_2} = ker(F)^{\Delta_2} = G_{B'}^{\Delta_2}$, as desired. □

Proposition 2.4 and proposition 2.6 describe the first two steps of the algorithm. Moreover, they prove that any partial base constructed by the execution of the first two steps of the algorithm can be extended to a minimum base. The third step of the algorithm uses Lovász's result for finding a maximum matching of a 2-polymatroid.

The polymatroid matching problem (also known as the matroid parity and the matchoid problem) is a generalization of the maximum matching problem for graphs and the matroid intersection problem. Lovász proved that the polymatroid problem is poly-

nomially unsolvable in general but solvable in polynomial time for linear matroids [31].

In a subsequent paper Lovász generalized his algorithm to handle a larger class of polymatroids [32]. We shall need the generalized matching algorithm to perform step 3. The following definitions and remarks are taken from the later paper.

Let S be a finite set and f an integer valued function defined on the subsets of S such that

$$f(\emptyset) = 0$$

$$X \subseteq Y \Rightarrow f(X) \leq f(Y)$$

$$f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y).$$

The pair (S, f) is called a polymatroid. If $f(x) \leq k$ for $x \in S$, then we call (S, f) a k-polymatroid. In this paper we assume that all polymatroids are 2-polymatroids. If the elements of S are subsets of a linear space and the function $f(X)$ is the dimension of the space spanned by the elements in X, then we call (S, f) a linear polymatroid.

A subset $X \subseteq S$ is called a matching if $f(X) = 2|X|$. We say that $X \subseteq S$ is a circuit if $f(X) = 2|X| - 1$ and for every $x \in X$ we have $f(X - \{x\}) = 2|X| - 1$. We call $X \subseteq S$ a double circuit if $f(X) = 2|X| - 2$ and for every $x \in X$ we have $f(X - \{x\}) = 2|X| - 3$.

Every double circuit has a unique partition $X = X_1 \cup \cdots \cup X_m$ $(m \geq 2)$, such that $X - X_i$ is a circuit for $1 \leq i \leq m$ and these are all the circuits contained in X. We call this partition the principal partition of the double circuit X. The double circuit is trivial if $m = 2$. We say that the "projection" (S, f') compresses the double circuit X if $f'(X - X_i) = f(X - X_i) - 1$ for $1 \leq i \leq m$.

If (S, f) is a linear 2-polymatroid in the linear space L, and X is a nontrivial double circuit in (S, r), then there is a vector $v \in L$, $v \neq 0$ which is contained in the linear span of each circuit $X - X_i$. Projecting everything onto a hyperplane complementary to p, we get a projection compressing X. Lovász points out that the only step of the maximum matching algorithm for linear 2-polymatroids that does not generalize to all 2-polymatroids is the construction of the projection described above [32, page 212].

To perform step (3) of the minimum base algorithm we construct a 2-polymatroid (S, f), where S is a collection of subsets taken from a direct product of two linear spaces. The following lemma proves that we can find a maximum matching for (S, f) in polynomial time using Lovász's algorithm.

Lemma 2.7 Let $L = (Z_2)^r \times (Z_3)^s$. Let f be the integer valued function defined on the subsets of L such that, $f(X) = r_1 + s_1$ if and only if $\langle X \rangle \simeq (Z_2)^{r_1} \times (Z_3)^{s_1}$.

Let S be a collection of two element subsets of L such that $f(x) = 2$, for each $x \in S$. Then (S, f) is a 2-polymatroid, and a maximum matching for (S, f) can be found using Lovász's algorithm.

Proof: It may be convenient to think of L as the direct product of two vector spaces and to view the elements of L as $(s + r)$-tuples. It is a simple exercise to check that (S, f) is a 2-polymatroid. Let $D \subset S$ be a nontrivial double circuit with principal partition $\{D_1, D_2, \ldots, D_m\}$ and define $K_i = D - D_i$ for $1 \leq i \leq m$.

To verify that a maximum matching can be found using Lovász's algorithm we must prove that we can find a nontrivial element of $K_1 \cap K_2 \cap \cdots \cap K_m$ in polynomial time. Using the principal partition for D, we can compute in polynomial time a set of generators

for $K_1 \cap K_2 \cap \cdots \cap K_m$.

To complete the proof we must guarantee that the intersection is nontrivial whenever the double circuit is nontrivial. Lovász uses an induction argument to prove that $K_1 \cap K_2 \cap \cdots \cap K_m$ is nonempty when the 2-polymatroid is linear [33, Lemma 11.3.3]. This same proof may be used for (S, f) by simply replacing each occurrence of the word, "dim" with the letter "f". □

Theorem 2.8 Let $G = \langle \Phi \rangle$ be an instance of AMB_7. We can compute a minimum base for G in polynomial time.

Proof: By Propositions 2.4-2.6 we may assume, without loss of generality, that all the G-orbits have size 4 or 6 and that $G \simeq (Z_2)^r \times (Z_3)^s$. For each G-orbit O_i, $1 \leq i \leq t$, we know that G^{O_i} is isomorphic to either $Z_2 \times Z_2$ or $Z_2 \times Z_3$. View each element $g \in G$ as a $2t$-tuple (i.e., $g = (g_1, g_2, \ldots g_{2t})$ where $g_i \in Z_2$ or $g_i \in Z_3$).

Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$, and define $\alpha_i = \phi_i^3$ and $\beta_i = \phi_i^2$ for $1 \leq i \leq m$. Then $\{\alpha_i, \beta_i | 1 \leq i \leq m\}$ is a generating set for G, such that $\langle \alpha_i | 1 \leq i \leq m \rangle = (Z_2)^r$ and $\langle \beta_i | 1 \leq i \leq m \rangle = (Z_3)^s$.

Define A be the $2m \times 2t$ matrix with rows $\alpha_1, \alpha_2, \ldots \alpha_m, \beta_1, \beta_2, \ldots \beta_m$. Let x_i be the i^{th} column of matrix A, $1 \leq i \leq 2t$. Then each x_i is a $2m$-tuple with $x_i \in (Z_2)^m \times (Z_3)^m$.

Let $X_i = \{x_{2i-1}, x_{2i}\}$ for $1 \leq i \leq t$, and let $S = \{X_1, \ldots, X_t\}$. Define the function f on the subsets of S such that $f(Y) = r_1 + s_1$ if and only if $\langle Y \rangle = (Z_2)^{r_1} \times (Z_3)^{s_1}$. By Lemma 2.7 we know that (S, f) is a 2-polymatroid and that we can find a maximum matching for (S, f) in polynomial time.

Observe that,

$$X = \{X_i | i \in I\} \text{ is a matching for } (S, f) \text{ if and only if } \Pi_{i \in I} G^{O_i} \leq G. \qquad \text{(II.1)}$$

Let $X = \{X_i | i \in I\}$ be a maximum matching for (S, f). For each $i \in I$ select a point $b_i \in O_i$, and let B be the partial base for G comprised of the points b_i. Since X is a maximum matching it follows that all the G_B-orbits have size 2 or 3. We can use the Greedy algorithm to extend B to a nonredundant base for G of size $s + r - |I|$.

To finish the proof it will suffice to prove that $\mathcal{M}(G) = s + r - |I|$. Suppose that there exists a base $A = a_1, a_2, \ldots, a_k$ for G, such that $k < s + r - |I|$. Let $G_i = G_{\{a_1, \ldots, a_i\}}$ for $1 \leq i \leq k$, and define J so that $[G_{j-1} : G_j]$ is nonprime if and only if $j \in J$.

Then $k < s + r - |I|$ implies that $|J| > |I|$. If $a_j \in O'_j$ for $j \in J$, then $\Pi_{j \in J} G^{O_{j'}} \leq G$ ($[G_{j-1} : G_j]$ is nonprime). By (II.1) this implies that $\{X_{j'} | j \in J\}$ is a matching for (S, f) and this contradicts the fact that $\{X_i | i \in I\}$ is a maximum matching. □

Remark 2.9 Let G be an abelian permutation group for which all the orbits have size a prime or a product of two primes; then we can find a minimum base for G in polynomial time.

Proof: For any G-orbit O, we know that the constituent G^O is abelian and transitive. Thus, G^O is regular and $|G^O| = |O|$. We can modify Propositions 2.4 and 2.5 so that the orbits of prime order can be ignored. Proposition 2.6 is essentially the same except that $F(G)$ is abelian semisimple (the $F(G)$-orbits have prime order). We generalize Lemma 2.7 to handle polymatroids (S, f), where S is a direct product of (possibly more than two) linear spaces. □

Remark 2.10 The problem of finding a maximum matching of a linear 2-polymatroid over the field $GF(p)$ is polynomial time equivalent to the problem of finding a minimum base for an abelian semisimple group G with orbits of size p^2.

Proof: One direction of the remark is proved by Theorem 2.8. To prove the other direction we assume, without loss of generality, that the linear 2-polymatroid (S, f) is specified by the columns of a $m \times 2r$ matrix M over the field $GF(p)$. Mimicking the proof of Theorem 2.8 we use the rows of matrix M to generate a group G with r orbits of size p^2. The remark follows from equation II.1. □

CHAPTER III

SHARP BOUNDS FOR BASES

We now know that the Greedy Algorithm cannot be used to solve the minimum base problem. In fact, unless P=NP there is no polynomial time solution to the minimum base problem. It is natural to ask whether the Greedy Algorithm is a good approximation heuristic for computing a small base. To answer this question we give, in the first section, a sharp bound for the size of a nonredundant base. This bound may then be compared to the result found in section 2, a sharp bound for the size of a greedy base.

In the last section we analyze a second greedy heuristic for approximating a minimum base. In some cases the new greedy heuristic outperforms our original Greedy algorithm. In other cases the old Greedy algorithm produces a smaller base than the new algorithm. To compare the two algorithms we consider the worst case performance of each.

A Sharp Bound for Nonredundant Bases

Babai has pointed out that the variation in size between two nonredundant bases for a group $G \leq \mathrm{Sym}(n)$ can be no more than a factor of $\log n$ [2].

Lemma 3.1 Let $G \leq \mathrm{Sym}(n)$, and let r be the size of any nonredundant base for G, then $r \leq \mathcal{M}(G) \log n$.

Proof: Fact 1.1 and Fact 1.3 imply that $2^r \leq |G| \leq n^{\mathcal{M}(G)}$. □

The fact that this bound is sharp is a consequence of the following lemma.

Lemma 3.2 Fix $k \geq 1$, then for any $n \geq 8k^2$ there exists $G \leq \mathrm{Sym}(n)$ such that

- $\mathcal{M}(G) = k$, and
- G has a nonredundant base of size at least $\frac{1}{3}\mathcal{M}(G)\log n$.

Proof: Suppose that $n \geq 8k^2$, and that r is the integer maximal with respect to $k2^r + 2rk \leq n$. Define $X = \{1, 2, \ldots, rk\}$, $X_i = \{r(i-1)+1, \ldots, ir\}$, and $Y_j = \{j\}$ for $1 \leq i \leq k$, $1 \leq j \leq rk$. Let $\mathcal{C} = \{X_1, \ldots, X_k, Y_1, \ldots, Y_{rk}\}$.

Using the notation from Chapter I we define $G = G_{\mathcal{C}}(X) \leq \mathrm{Sym}(\Omega_{\mathcal{C}})$. Then $G \simeq (Z_2)^{rk}$, and $|\Omega_{\mathcal{C}}| = k2^r + 2rk$.

Since a largest G-orbit has size 2^r and $|G| = 2^{kr}$ it follows from Facts 1.1 and 1.3 that a minimum base for G must have size at least k. Let $A = a_1, a_2, \ldots a_k$ where $a_i \in G(X_i)$, then by Lemma 1.5 it follows that A is a minimum base for G.

Now we show that the group G has a nonredundant base of size at least $\frac{1}{3}k\log n$. Let $B = b_1, b_2, \ldots, b_{rk}$ where $b_j \in G(Y_j)$, and let $G = G^0 \geq G^1 \geq \cdots \geq G^{rk}$ be the chain of stabilizers of G relative to B. By Lemma 1.5 we have $G^i = G_{\mathcal{C}}(X \setminus \{1, 2, \ldots, i\})$, and it follows that B is a nonredundant base for G. The size of B is $rk \geq \frac{1}{3}k\log n$, since $k2^{r+2} \geq n \geq 8k^2$. □

If B is any nonredundant base $G \leq Sym(n)$, then $\mathcal{M}(G) \leq |B| \leq \mathcal{M}(G)\log n$. We know that there exist groups that have nonredundant bases as large as $\mathcal{M}(G)\log n$. In the next section we show that if B is a greedy base for G, then $|B|$ is closer to $\mathcal{M}(G)$ than to $\mathcal{M}(G)\log n$.

A Sharp Bound for Greedy Bases

In contrast to the $\log n$ indeterminacy of an arbitrary nonredundant base we show that a greedy base is within a $\log\log n$ factor of optimal.

Lemma 3.3 If $G \leq \mathrm{Sym}(n)$ has a base of size k, then there exists a G-orbit of size at least $|G|^{\frac{1}{k}}$.

Proof: Follows from Fact 1.1 and Fact 1.3. □

Theorem 3.4 If $G \leq \mathrm{Sym}(n)$, then any greedy base for G has size no more than $\lceil \mathcal{M}(G) \log\log n \rceil + \mathcal{M}(G)$.

Proof: Let $B = b_1, b_2, \ldots, b_m$ be a greedy base for G, and $G = G^0 \geq G^1 \geq \cdots \geq G^m = \{id\}$ be the chain of stabilizers of G relative to B. First we show that

$$|G^i| \leq |G|^{((\mathcal{M}(G)-1)/\mathcal{M}(G))^i}, 0 \leq i \leq m. \tag{III.2}$$

The statement holds trivially when $i = 0$. Assuming the statement is true for i, we show that it is true for $i+1$. Since $\mathcal{M}(G^i) \leq \mathcal{M}(G)$ it follows from Lemma 3.3 that G^i must have an orbit of size at least $|G^i|^{\frac{1}{\mathcal{M}(G)}}$. Since b_{i+1} was chosen via the Greedy Algorithm we know that $|G^i : G^{i+1}|$ is the size of a largest G^i-orbit. Thus $|G^i : G^{i+1}| \geq |G^i|^{1/\mathcal{M}(G)}$, and this implies that $|G^{i+1}| \leq |G^i|^{(\mathcal{M}(G)-1)/\mathcal{M}(G)} \leq |G|^{((\mathcal{M}(G)-1)/\mathcal{M}(G))^{i+1}}$.

We observe next that

$$i = \lceil \mathcal{M}(G) \log \frac{\log |G|}{\mathcal{M}(G)} \rceil \Rightarrow |G^i| \leq 2^{\mathcal{M}(G)}. \tag{III.3}$$

To see this, note that $\mathcal{M}(G) \geq \frac{1}{\log \frac{\mathcal{M}(G)}{\mathcal{M}(G)-1}}$ for $\mathcal{M}(G) \geq 2$. So, $i = \lceil \mathcal{M}(G) \log \frac{\log |G|}{\mathcal{M}(G)} \rceil$ implies $|G|^{((\mathcal{M}(G)-1)/\mathcal{M}(G))^i} \leq 2^{\mathcal{M}(G)}$, which, by (III.2), implies $|G^i| \leq 2^{\mathcal{M}(G)}$.

If the group G^i has order less than or equal to $2^{\mathcal{M}(G)}$ then the size of any nonredundant base for G^i is at most $\mathcal{M}(G)$. Now, combining (III.3) with the fact that the Greedy Algorithm produces a nonredundant base, we have

$$m \leq \lceil \mathcal{M}(G) \log \frac{\log |G|}{\mathcal{M}(G)} \rceil + \mathcal{M}(G).$$

By Lemma 3.3 we have $n \geq |G|^{\frac{1}{\mathcal{M}(G)}}$, implying $\log |G| \leq \mathcal{M}(G) \log n$ and the result follows. □

In proving the bound is sharp we use the following technical lemma.

Lemma 3.5 Let r, k be two positive integers such that $r \geq k \geq 2$. Define $r_0 = r$ and $r_i = r_{i-1} - \lfloor r_{i-1}/k \rfloor - 1$ for $i \geq 1$. If $\gamma = \lfloor (k/2)(\log(r+k) - \log(k+1)) \rfloor$, then $r_\gamma \geq 1$.

Proof: Define $s_0 = r$ and $s_i = s_{i-1} - \frac{s_{i-1}}{k} - 1$ for $i \geq 1$. By a straightforward inductive argument, we have $s_i \leq r_i$ and $s_i = (1 - \frac{1}{k})^i r - k(1 - (1 - \frac{1}{k})^i)$ for $i \geq 0$. Then

$$s_i \geq 1 \Leftrightarrow i \leq \frac{\log(r+k) - \log(k+1)}{\log \frac{k}{k-1}}.$$

The result follows since $\frac{k}{2} \leq \frac{1}{\log \frac{k}{k-1}}$. □

Theorem 3.6 Fix $k \geq 2$, then for any $n \geq 2^{4k^2+7k+7}$ there exists $G \leq \mathrm{Sym}(n)$ such that

- $\mathcal{M}(G) = k$, and
- Every greedy base for G has size at least $\frac{1}{5}\mathcal{M}(G) \log \log n$.

Proof: Suppose that $n \geq 2^{4k^2+7k+7}$, and that r is the largest integer such that $k2^r + 2^{r+k+1} \leq n$. Let X be a set of order rk. The set X is partitioned into k sets of order r, $A_{1,0}, A_{2,0}, \ldots, A_{k,0}$. We now recursively define sets $A_{i,j}$ for $1 \leq i \leq k$, and $1 \leq j \leq \gamma$ (γ defined later) as follows: $A_{i,j}$ is a subset of $A_{i,j-1}$ created by removing $\lfloor \frac{|A_{i,j-1}|}{k} \rfloor + 1$ elements from $A_{i,j-1}$. The elements removed from the k sets $A_{1,j-1}, A_{2,j-1}, \ldots, A_{k,j-1}$ are placed in a set B_j. Note that $|B_j| > |A_{i,j-1}|$. Let $\gamma = \lfloor (k/2)(\log(r+k) - \log(k+1)) \rfloor$. The value of γ was computed in Lemma 3.5 to insure that $|A_{i,j}| \geq 1$ for all values of i and j. Let $\mathcal{C} = \{A_{1,0}, \ldots, A_{k,0}, B_1, \ldots, B_\gamma\}$.

Once again we use the notation from Chapter I to define $G = G_{\mathcal{C}}(X) \leq \mathrm{Sym}(\Omega_{\mathcal{C}})$. Recall that $G \simeq (Z_2)^{rk}$, and $|\Omega_{\mathcal{C}}| = k2^r + 2^{|B_1|} + 2^{|B_2|} + \ldots + 2^{|B_\gamma|}$.

Consider the sequence of points $A = a_1, a_2, \ldots, a_k$ where $a_i \in G(A_{i,0})$. Since $X = \dot{\bigcup}_{i=1}^{k} A_{i,0}$, it follows from Lemma 1.5 that A is a base for G. Moreover, any subcollection $\mathcal{C}'$ of $\mathcal{C}$ that covers X (i.e., $X = \bigcup_{Y \in \mathcal{C}'} Y$) must contain all the $A_{i,0}$, $1 \leq i \leq k$. It follows that A is a minimum base for G.

Next we show that the Greedy Algorithm must select $B = b_1, b_2, \ldots, b_\gamma$ as a partial base for G, where $b_j \in G(B_j)$. It suffices to show that $G(B_j)$ is the largest G^{j-1}-orbit. By Lemma 1.5 we have $G^{j-1} = G_{\mathcal{C}}(X \setminus \dot{\bigcup}_{l=1}^{j-1} B_l) = G_{\mathcal{C}}(\dot{\bigcup}_{i=1}^{k} A_{i,j-1})$.

Now using Lemma 1.6 we see that the points in $G(A_{i,0})$ are partitioned into G^{j-1}-orbits of size $2^{|A_{i,j-1}|}$ for $1 \leq i \leq k$. The action of G^{j-1} on points $G(B_i)$ is trivial if $1 \leq i \leq j-1$ and transitive if $j \leq i \leq \gamma$. Thus b_j is in a G^{j-1}-orbit of size $2^{|B_j|}$, and this is the largest G^{j-1}-orbit.

So far we have shown that G has a minimum base of size k, and that the Greedy Algorithm produces a base of size at least γ. To finish the proof we note that $\gamma \geq$

$\frac{1}{4}\mathcal{M}(G)\log r$ and $r \geq \frac{1}{2}\log n$. □

Another Greedy Heuristic

A student at Oxford, Tracey Maund, suggested a different greedy heuristic for computing small bases. Instead of fixing a point whose stabilizer has the smallest order (i.e., a point in a largest orbit), choose a point whose stabilizer has the largest number of orbits (including trivial orbits). To avoid any confusion, call the new greedy algorithm Greedy2 and the original greedy algorithm Greedy1.

Definition 3.7 For any permutation group G define a function ρ from G into the natural numbers such that $\rho(G)$ is equal to the number of G-orbits.

We can modify Jerrum's algorithm to compute a Greedy2 base for $G \leq Sym(n)$ in $O(n^5)$ time. First, note that if $H \leq G$ and b, b' are in the same H-orbit, then $\rho(G_b) = \rho(G_{b'})$. Thus, in each iteration of Jerrum's algorithm it will suffice to consider just one point b from each G^{i-1}-orbit. For each b we compute a set of Schreier generators for G_b^{i-1} and select the b that maximizes $\rho(G_b^{i-1})$. It takes $O(n^4)$ time to select b_i, and hence the Greedy2 algorithm runs in $O(n^5)$ time.

It was observed that in several cases the Greedy2 algorithm produced a smaller base than the Greedy1 algorithm. In particular, if we consider $Sym(m)$ acting on the set of pairs of $\{1, 2, \ldots, m\}$, then for m sufficiently large a Greedy2 base for G is smaller than a Greedy1 base for G [8].

We can, in fact, use a construction similar to the one found in the proof of Theorem 2.2 to manufacture groups for which any Greedy2 base is larger than any Greedy1 base and groups for which any Greedy2 base is smaller than any Greedy1 base. The following

two examples illustrate this point.

Example 3.4 Let $X = \{x_1, x_2, x_3, y_1, y_2\}$, $Y_1 = \{x_1, x_2, x_3\}$, $Y_2 = \{y_1, y_2\}$, and $Y_3 = Y_4 = Y_5 = \{x_1, y_1\}$. Define $\mathcal{C} = \{Y_1, Y_2, Y_3, Y_4, Y_5\}$, and let $G = G_{\mathcal{C}}(X)$. Recall that $G \simeq (Z_2)^5$, and that $\Omega_{\mathcal{C}} = \dot{\bigcup}_{i=1}^{5} G(Y_i)$.

Since each $G(Y_i)$ is a G-orbit it follows from Fact 1.1 that any base for G must have size at least 2. Using Lemma 1.5 and Lemma 1.6 we see that $B = b_1, b_2$, where $b_i \in G(Y_i)$ for $i = 1, 2$ is a Greedy1 base for G.

On the other hand the Greedy2 algorithm will always start by fixing a point in either $G(Y_3)$, $G(Y_4)$ or $G(Y_5)$, resulting in a base of size 3.

Example 3.5 Let $X = \{x_1, x_2, x_3, y_1, y_2, y_3\}$, $Y_1 = \{x_1, x_2, x_3\}$, and let $Y_2 = \cdots = Y_7 = \{x_1, y_1\}$. Define $Y_8 = Y_9 = Y_{10} = \{x_2, y_2\}$, and $Y_{11} = \{x_3, y_3\}$. Let $\mathcal{C} = \{Y_1, Y_2, \ldots Y_{11}\}$ and let $G = G_{\mathcal{C}}(X)$.

Then $\Omega_{\mathcal{C}} = \dot{\bigcup}_{i=1}^{k} G(Y_i)$, and each $G(Y_i)$ is a G-orbit. Using Lemma 1.5 one checks that $B = b_1, b_2, b_3$ is a minimum base for G, where $b_1 \in G(Y_2)$, $b_2 \in G(Y_8)$ and $b_3 \in G(Y_{11})$. Moreover, the Greedy2 algorithm will always construct a base of size 3.

In contrast, the Greedy1 algorithm will start by fixing a point in the G-orbit $G(Y_1)$, and thus construct a base of size 4.

We know that specific examples cannot be used to compare the two greedy heuristics. Instead, we shall use the worst case performance as a means of comparison. We already have a sharp bound for the worst case performance of the Greedy1 algorithm. What we need now is a sharp bound for the worse case performance of the Greedy2 algorithm. Unfortunately, we are unable to find such a bound. In lieu of a sharp bound, we show

that for n sufficiently large there exists $G \leq Sym(n)$ such that, any Greedy2 base for G is arbitrarily close to the upper bound $O(\mathcal{M}(G)\log n)$.

Theorem 3.8 Fix $k \geq 2$ and $0 < \epsilon < 1$. Let N be the smallest integer for which $(\log N)^{\epsilon} \geq \log\log N$. For any integer $n \geq max(N, 2^{2^{k+1}})$ there exists $G \leq Sym(n)$ such that

- $\mathcal{M}(G) = k$, and
- Every Greedy2 base for G has size at least $\frac{1}{6}\mathcal{M}(G)(\log n)^{1-\epsilon}$.

Proof: Let $n \geq max(N, 2^{2^{k+1}})$ and let r be the integer maximal with respect to $k2^r + 2^{r+1}\lfloor \frac{r}{c} \rfloor \leq n$, where $c = \lceil \frac{\log r}{k-1} \rceil$. Let X be a set of order rk, and partition X into k sets, $A_1, A_2, \ldots, A_k$ each of size r.

Define sets $B_{i,1}$, $1 \leq i \leq \lfloor \frac{r}{c} \rfloor$, so that, $B_{i,1}$ contains exactly c elements from each of the A_j ($|B_{i,1}| = ck$), and so that all the $B_{i,1}$ are disjoint. Let $B_{i,j} = B_{i,1}$ for $1 \leq i \leq \lfloor \frac{r}{c} \rfloor$ and $2 \leq j \leq 2^{r-ck+1}$, and define

$$\mathcal{C} = \{A_l, B_{i,j} | 1 \leq l \leq k,\ 1 \leq i \leq \lfloor \frac{r}{c} \rfloor,\ 1 \leq j \leq 2^{r-ck+1}\}.$$

Using notation from Chapter I we let $G = G_{\mathcal{C}}(X)$. Recall that $G \leq \mathrm{Sym}(\Omega_{\mathcal{C}})$ and that $G \simeq (Z_2)^{rk}$.

First we observe that $G \leq Sym(n)$, since $|\Omega_{\mathcal{C}}| = k2^r + 2^{r+1}\lfloor \frac{r}{c} \rfloor$. Next we show that $\mathcal{M}(G) = k$. To see this consider the sequence of points $A = a_1, a_2, \ldots, a_k$ where $a_i \in G(A_i)$. Since $X = \dot{\bigcup}_{i=1}^{k} A_i$, it follows from Lemma 1.5 that A is a base for G. Furthermore, since all of the G-orbits have size no more than 2^r, Facts 1.1 and 1.3 imply that A is a minimum base for G.

To finish the proof we must show that condition (2) holds. Consider the sequence $B = b_1, b_2, \ldots, b_{\lfloor \frac{r}{c} \rfloor}$, where $b_i \in \dot{\bigcup}_{j=1}^{2^{r-ck+1}} G(B_{i,j})$. Let $G^0 > G^1 > \cdots > G^{\lfloor \frac{r}{c} \rfloor}$ be the chain of stabilizers of G relative B (B is not necessarily a base for G). By Lemma 1.5 and Lemma 1.6 we have,

$$\rho(G_a^{i-1}) = m + 2^r + (k-1)2^{c(i-1)} + (i-1)2^{r+1} + (\lfloor \frac{r}{c} \rfloor - (i-1))2^{r-ck+1+c} \text{ and}$$

$$\rho(G_{b_i}^{i-1}) = m + k2^{ic} + i2^{r+1} + (\lfloor \frac{r}{c} \rfloor - i)2^{r-ck+1},$$

where $m = n - |\Omega_C|$, $1 \leq i \leq \lfloor \frac{r}{c} \rfloor$ and $a \in G(A_j)$ for $1 \leq j \leq k$.

To prove that B is a partial Greedy2 base we must show that $\rho(G_a^{i-1}) < \rho(G_{b_i}^{i-1})$. Observe that $2^{2^{k+1}} \leq n \leq 2^{2r}$ implies that $2 \leq c$. Since $2 \leq c$ it follows that $2^r \geq (\lfloor \frac{r}{c} \rfloor - (i-1))2^{r-ck+1+c}$, and this implies that $\rho(G_a^{i-1}) < \rho(G_{b_i}^{i-1})$. Furthermore, by the construction of the $B_{i,j}$ we may conclude that any Greedy2 base for G will have size at least $\lfloor \frac{r}{c} \rfloor$.

To finish the proof we must prove that $\lfloor \frac{r}{c} \rfloor \geq \frac{1}{6}\mathcal{M}(G)\log n^{1-\epsilon}$. Note that $r \leq \log n < 2r$ implies that $\frac{1}{2}(\log n)^{\epsilon}(\log n)^{1-\epsilon} < r$ and that $\log r \leq \log\log n$. Now the result follows from the fact that $N \leq n$ and $k \leq \log r$. □

Theorem 3.4 and Theorem 3.8 allow us to compare the worst case performance of algorithms Greedy1 and Greedy2.

CHAPTER IV

P-COMPLETE ALGEBRAIC PROBLEMS

We say that a decision problem is in NC if there exists a PRAM algorithm for the problem that runs in time $O((\log n)^{c_1})$ using $O(n^{c_2})$ processors for constants c_1 and c_2 [14]. A decision problem A is logspace reducible to a decision problem B, if there is a function f, computable by a logspace Turing machine, that satisfies the property that $x \in A$ if and only if $f(x) \in B$.

A decision problem in P is P-complete if every decision problem in P is logspace reducible to it. Note that logspace reducibility is transitive, and has the following two properties. First, if A is logspace reducible to B and B is in NC, then A is in NC. Second, if A is logspace reducible to B and A is P-complete, then B is P-complete (provided that B is in P).

Every decision problem in NC lies in P. A fundamental problem in complexity theory is whether P=NC. If this were the case it would mean that every problem in P is parallelizable (i.e., can be solved efficiently in parallel). Under the assumption that P$\neq$NC, a statement which most computer scientists believe to be true, we would like to identify the problems that are in NC and the problems that are in P\NC.

To show that a problem is in NC it suffices to describe a PRAM algorithm for the problem that runs in polylog time and uses no more than a polynomial number of

processors. To show that a problem is in P\NC (assuming that P≠NC) we reduce a P-complete problem to it. The standard P-complete problem used for this purpose is the Monotone Circuit Value Problem (MCVP).

MCVP Input: A set of boolean functions $g_0, g_1, \ldots, g_n$, where $g_0 = 0$, $g_1 = 1$ and for $2 \leq i \leq n$, g_i is equal to either $g_j \wedge g_k$ or $g_j \vee g_k$, where $j, k < i$.

Question: Does $g_n = 1$?

In this chapter we prove that the two algebraic problems, deterministic greedy base and factoring (both described later), are P-complete. This is done by reducing a restricted version of the P-complete problem, greedy independent set (GIS), to our algebraic problems. We sketch Cook's proof that GIS is P-complete, and point out why the restricted version of the problem remains P-complete. We conclude the chapter with a PRAM algorithm that proves that factoring is in NC for solvable groups.

Greedy Bases and Independent Sets

Let $\Gamma(V, E)$ be a graph with vertex set V and edge set E. A subset $W \subseteq V$ is called an independent set of vertices in $\Gamma(V, E)$, if for all $w_1, w_2 \in W$, $(w_1, w_2) \notin E$.

There is a natural greedy algorithm for constructing a maximal independent set of vertices in $\Gamma(V, E)$. Given a linear ordering of the vertex set V, the greedy algorithm repeatedly picks the smallest vertex from V that is not adjacent to a previously selected vertex. The corresponding decision problem, greedy independent set, is defined as follows:

GIS Input: Graph $\Gamma(V, E)$ where V is linearly ordered.

Question: Is the last vertex in the ordering part of the greedy maximal independent set?

Proposition 4.1 [Cook] The GIS problem is P-complete.

Proof: The GIS problem is clearly in P. To prove the problem is complete we sketch Cook's logspace reduction of MCVP to GIS.

Let $g_0, g_1, \ldots g_n$ be an instance of the MCVP. We construct a graph $\Gamma(V, E)$ with vertex set $V = \{v_0, v_1, \ldots, v_n\} \cup \{w_0, w_1, \ldots, w_n\}$. We order the vertices so that v_i and w_i precede v_j and w_j, whenever $i < j$. The ordering of v_i relative to w_i is determined by the gate g_i. Let w_0 precede v_0 and let v_1 precede w_1. For any i, $2 \leq i \leq n$, w_i precedes v_i if $g_i = g_j \vee g_k$, and v_i precedes w_i if $g_i = g_j \wedge g_k$. This gives us a linear ordering of the set V.

The edge set, E, is equal to $E_1 \cup E_2 \cup E_3$, where

$$
\begin{aligned}
E_1 &= \{(v_i, w_i) | 0 \leq i \leq n\}, \\
E_2 &= \{(w_i, v_j), (w_i, v_k) | 2 \leq i \leq n \text{ and } g_i = g_j \vee g_k\} \text{ and} \\
E_3 &= \{(v_i, w_j), (v_i, w_k) | 2 \leq i \leq n \text{ and } g_i = g_j \wedge g_k\}.
\end{aligned}
$$

Note that the construction of $\Gamma(V, E)$ from the instance of the MCVP can be performed by a logspace algorithm. A simple induction argument shows that v_i is in the greedy independent set for $\Gamma(V, E)$ if and only if $g_i = 1$, and w_i is in the greedy independent set for $\Gamma(V, E)$ if and only if $g_i = 0$. □

Remark 4.2 Let $\Gamma(V, E)$ be an instance of the GIS problem where the linear ordering of V is $v_0 < v_1 < \cdots < v_n$. The GIS problem remains P-complete even if we restrict ourselves to instances in which the following conditions are true. We assume that $(v_0, v_1) \in E$ and we assume that v_i, $2 \leq i \leq n$, is connected to exactly two vertices that are smaller than itself.

Proof: It suffices to note that the following changes can be made to the reduction of MCVP to GIS. First, we may assume without loss of generality, that if $g_i = g_j \vee g_k$ (or $g_i = g_j \wedge g_k$) and $2 \leq i \leq n$, then $j \neq k$. Second, we may eliminate node v_0 from the construction of $\Gamma(V, E)$, and we may add edge (w_0, w_1) to E. All the nodes in the set $X = \{v_i, w_i | 2 \leq i \leq n\}$ are connected to either 1 or 2 nodes smaller than themselves. For any node $x \in X$ connected to only 1 node smaller than itself, add the edge (x, w_1) to E. □

To define a greedy base decision problem it is necessary to modify the Greedy (base) Algorithm so that it is deterministic. During the i^{th} step of the original algorithm we have a choice of picking any point from any largest G^{i-1}-orbit. We can eliminate this nondeterminism by ordering the points and insisting that we always pick the smallest eligible point.

Let us call this the deterministic greedy (base) algorithm. With respect to this algorithm we can talk about "the" greedy base for a group. Note that with respect to the original Greedy Algorithm there could be an exponential number of greedy bases for a group. We define the deterministic greedy base (DGB) problem as follows:

DGB Input: A generating set for $G \leq \mathrm{Sym}(\Omega)$, a linear ordering of Ω and a fixed $\omega \in \Omega$.

Question: Is ω part of the greedy base for G?

Lemma 4.3 The DGB problem is P-complete.

Proof: Since the deterministic greedy base is unique we know that the DGB problem is in P. To prove that the problem is P-complete it will suffice to show that there is a logspace reduction of GIS to DGB.

Let $\Gamma(V, E)$ be an instance of GIS with linear ordering $v_0 < v_1 < \cdots < v_n$. By Remark 4.2 we may assume, without loss of generality, that $(v_0, v_1) \in E$ and that each v_i, $2 \leq i \leq n$, is connected to exactly two smaller vertices.

Let $X = \{v_0, v_1, \ldots, v_n\} \cup \{w_1, w_2, \ldots w_n\}$ and let $W_i = \{w_i, v_i\}$ for $1 \leq i \leq n$. Define $Y_0 = Y_1 = \{v_0, w_1, v_1\}$ and for i, $2 \leq i \leq n$, let $Y_i = \{v_i, v_j, v_k\}$, where v_j, v_k are the two unique vertices less than v_i that are connected to v_i.

Let $\mathcal{C} = \{W_i, Y_j | 1 \leq i \leq n, 0 \leq j \leq n\}$ and let $G = G_{\mathcal{C}}(X) \leq Sym(\Omega_{\mathcal{C}})$. Recall that $\Omega_{\mathcal{C}}$ is the disjoint union of the sets $G(W_i)$ and $G(Y_j)$, $1 \leq i \leq n$, $0 \leq j \leq n$, and that G is generated by the permutations $\{\mathcal{R}_{\mathcal{C}}(\sigma_x) | x \in X\}$.

Order the elements in each set $G(W_i)$ and $G(Y_j)$, $1 \leq i \leq n$, $0 \leq j \leq n$ arbitrarily. We will extend this to a linear ordering on $\Omega_{\mathcal{C}}$ by ordering the sets so that,

$$G(Y_0) < G(W_1) < G(W_2) < \cdots < G(W_n) < G(Y_1) < G(Y_2) < \cdots < G(Y_n).$$

Let b_i be the smallest point in $G(Y_i)$. We define an instance of the DGB problem

where $G = \langle \mathcal{R}_{\mathcal{C}}(\sigma_x) | x \in X \rangle$, and the ordering of the set is defined as above. We wish to know if $b_n \in \Omega_{\mathcal{C}}$ is in the deterministic greedy base.

Since G has degree $4n + 8(n+1)$ and is specified by $2n+1$ generators, it follows that this instance of DGB can be constructed from $\Gamma(V, E)$ by an algorithm that uses no more than $O(\log n)$ space. To finish the proof we must show that v_n is in the greedy independent set if and only if b_n is in the deterministic greedy base.

Let B' be the set of points selected by the deterministic greedy base algorithm, and let V' be the greedy independent set. We shall prove by induction on i, $0 \leq i \leq n$ that $v_i \in V'$ if and only if $b_i \in B'$.

Clearly the hypothesis is true for $i = 0$ and $i = 1$. Assume the statement is true for all i, $1 < i < m$, and let v_j and v_k be the two vertices less than v_m that are connected to v_m. Then $v_m \in V'$ if and only if $v_j \notin V'$ and $v_k \notin V'$. By the induction hypothesis this implies that $v_m \in V'$ if and only if $b_j \notin B'$ and $b_k \notin B'$. Note that $b_j \notin B'$ and $b_k \notin B'$ implies that $b_m \in B'$, since $|G(Y_m)| = 8$ and $|G(W_m)| = |G(W_j)| = |G(W_k)| = 4$. If b_i or b_k is in B' then the set $G(W_m)$ guarantees that $b_i \notin B'$. Thus, $v_m \in V'$ if and only if $b_m \in B'$. □

Factoring with an SGS is P-complete

We now turn our attention to the factoring problem. Given a base and an SGS for $G \leq Sym(n)$, can we factor (i.e., sift) $g \in G$ in parallel through the SGS? This was stated as an open problem in [4]. We show that such an algorithm exists if and only if P=NC. We define the factoring problem as follows:

FAC Input: An SGS, $U = \bigcup_{i=1}^{k} U_i$, for $G \leq Sym(n)$ relative to a base $B = b_1, b_2, \ldots b_k$. An element $g \in G$, and $u \in U_k$.

Question: Does $g = u_k u_{k-1} \cdots u_1$ where $u_i \in U_i$ and $u_k = u$?

Lemma 4.4 FAC is P-complete.

Proof: Since sifting takes $O(nk)$ time it follows that FAC is in P. To show that FAC is P-complete we describe a logspace reduction of GIS to FAC.

Let $\Gamma(V, E)$ be an instance of GIS, with linear ordering $v_1 < v_2 < \cdots < v_n$. We may assume, without loss of generality, that $(v_1, v_2) \in E$ and that each v_i, $2 \leq i \leq n$, is connected to exactly two vertices less than itself.

Let $G < Sym(3n)$ generated by the 3-cycles $\{(3i-2, 3i-1, 3i) | 1 \leq i \leq n\}$. We view each element $g \in G$ an an n-tuple $g = (g_1, g_2, \ldots, g_n)$, where g_i is equal to either $\bar{0}, \bar{1}$ or $\bar{2}$ (i.e., g_i is equal to either id, $(3i-2, 3i-1, 3i)$ or $(3i-2, 3i, 3i-1)$).

Let $B = 3, 6, \ldots, 3n$ and $G_i = \{(0, \ldots, 0, g_{i+1}, \ldots, g_n) | g_j \in \{\bar{0}, \bar{1}, \bar{2}\}, i+1 \leq j \leq n\}$. Then $G = G^0 > G^1 > \cdots > G^n = \{id\}$ is the chain of stabilizers of G relative to B.

We define a set of coset representatives $U_i = \{\alpha_i, \beta_i, \gamma_i\}$ for G^i in G^{i-1}, $1 \leq i \leq n$, as follows. Set $\alpha_i = id$, $\beta_i = (0, \ldots, 0, g_i, 0, \ldots, 0)$ $g_i = \bar{1}$ and $\gamma_i = (h_1, h_2, \cdots, h_n)$, where

$$h_j = \begin{cases} \bar{2} & \text{if } j = i \\ \bar{1} & i < j \text{ and } (v_i, v_j) \in E \\ \bar{0} & \text{otherwise.} \end{cases}$$

Clearly $U = \bigcup_{i=1}^{n} U_i$ is an SGS for G relative to B, and the set U can be constructed from $\Gamma(V, E)$ by an algorithm that uses no more than $O(\log n)$ space. To complete the

construction we define $g = (\bar{2}, \bar{2}, \ldots, \bar{2}) \in G$ and $u = (\bar{0}, \bar{0}, \ldots, \bar{0}, \bar{2}) \in U_n$.

Let V' be the greedy maximal independent set for $\Gamma(V, E)$. It suffices to show that $v_n \in V'$ if and only if $g = u_n u_{n-1} \cdots u_1$, where $u_i \in U_i$ and $u = u_n$.

Fix m, $2 < m \leq n$, and suppose that $g = u_n u_{n-1} \cdots u_1$ and v_j, v_k are the two unique vertices less than v_m that are connected to v_m. Let

$$g u_1^{-1} u_2^{-1} \cdots u_{m-1}^{-1} = (0, \ldots, 0, h_m, h_{m+1} \ldots, h_n),$$

then by the definition of the SGS, U, it follows that

$$h_m = \bar{2} \text{ if and only if } u_j \neq \gamma_j \text{ and } u_k \neq \gamma_k. \tag{IV.4}$$

We prove by induction on i, $1 \leq i \leq n$ that $v_i \in V'$ if and only if $u_i = \gamma_i$. The hypothesis holds for $i = 1$ and $i = 2$. Assume that the statement is true for all i, $1 < i < m$. Let v_j, v_k be the two vertices less than v_m for which $(v_m, v_j), (v_m, v_k) \in E$. Then $v_m \in V'$ if and only if $v_j \notin V'$ and $v_k \notin V'$. From the induction hypothesis it follows that $v_m \in V'$ if and only if $u_j \neq \gamma_j$ and $u_k \neq \gamma_k$. Thus, by equation IV.4 $v_m \in V'$ if and only if $u_m = \gamma_m$. □

In the proof of Lemma 4.4 the group constructed for FAC was isomorphic to $(Z_3)^n$. We can find an SGS for this group relative to B for which factoring can be performed in NC (e.g., powers of the standard "vector space" basis). We will call such an SGS NC-efficient. If we assume that P $\neq$ NC, then from the above discussion it follows that some SGSs are NC-efficient and some are not. This brings up the question as to when there exists an NC-efficient SGS for fixed subgroup towers. Theorem 4.7 gives a partial

answer to this question.

Proposition 4.5 Let $H \leq G$ and let N be a group that is normalized by G (i.e., $gN = Ng$ for all $g \in G$). Let $A = \{a_1, a_2, \ldots, a_m\}$ be a complete set of coset representatives for HN in GN, such that $a_i \in G$ $1 \leq i \leq m$. If $B = \{b_1, b_2, \ldots, b_k\}$ is a complete set of coset representatives of $H \cap N$ in $G \cap N$, then $AB = \{ab | a \in A, b \in B\}$ is a complete set of coset representatives of H in G.

Proof: First observe that $[G : H] = [GN : HN][G \cap N : H \cap N]$. Thus, it will suffice to show that if $a_1, a_2 \in A$, $b_1, b_2 \in B$ and $a_1 b_1 H = a_2 b_2 H$, then $a_1 = a_2$ and $b_1 = b_2$.

Since $b_1, b_2 \in N$, $a_1 b_1 H = a_2 b_2 H$ implies that $a_1 = a_2$. Thus $b_1 H = b_2 H$, and this implies that $b_1^{-1} b_2 \in H$. □

Definition 4.6 A power-commutator basis (PCB) for a group G is an ordered sequence $(b_1, \rho_1), \ldots, (b_m, \rho_m)$, $b_i \in G$, $\rho_i > 1$ an integer, $1 \leq i \leq m$, such that:

(a) each $g \in G$ is uniquely expressible in "canonical form" $b_1^{e_1} \cdots b_m^{e_m}$, $0 \leq e_i < \rho_i$, $0 \leq i \leq m$

(b) for each pair of integers i, j, $1 \leq i < j \leq m$, the canonical expression for the commutator $[b_j, b_i]$ satisfies $e_1 = e_2 = \cdots e_i = 0$

(c) for each integer i, $1 \leq i \leq m$, the canonical expression for the element $b_i^{\rho_i}$ also satisfies $e_1 = e_2 = \cdots = e_i = 0$

Theorem 4.7 Suppose $G \leq Sym(n)$ is solvable and we are given generators for each subgroup in the polynomial tower $G = G_0 \geq G_1 \geq \cdots \geq G_m = \{id\}$, then we can find an NC-efficient SGS for the tower.

Proof: Using machinery established in [35] we can compute, in NC, a PCB for a normal tower $G = N_0 > N_1 > \cdots > N_k = \{id\}$, such that k is $O((\log n)^2)$ and N_{j-1} modulo N_j is abelian semisimple, $1 \leq j \leq k$.

Throughout this proof we view N_{j-1}/N_j as a direct product of vector spaces, and we shall refer to the N_{j-1}/N_j as vector spaces.

Luks and McKenzie introduce the notion of a "structure forest" and develop linear algebra techniques needed to find, in NC, the following:

(a) a homomorphism $\Phi_j : N_{j-1} \rightarrow Sym(A)$, such that $|A|$ is polynomial in n
and the $ker\Phi_j = N_j$, $1 \leq j \leq k$

(b) (PCB) elements in N_{j-1} that map to a basis of $\Phi_j(N_{j-1})$

Using the vector space representation, $\Phi_j(N_{j-1}) = N_{j-1}/N_j$, we solve, in NC, the factoring problem modulo N_j. The elements we find in $\Phi_j(N_{j-1})$ are then pulled back to inverse images in G. Next, we describe how the normal tower is used to "slice up" the groups in the tower $G = G_0 \geq G_1 \geq \cdots \geq G_m = \{id\}$.

Using results from [4] generators for $H_j^i = (G_i \cap N_{j-1})N_j$ can be found in NC. Note that this gives us a tower of subspaces for each j, $1 \leq j \leq k$,

$$\Phi_j(N_{j-1}) = \Phi_j(H_j^0) \geq \Phi_j(H_j^1) \geq \cdots \geq \Phi_j(H_j^m) = \{id\}. \qquad \text{(IV.5)}$$

Moreover, we can find in NC sets Γ_j^i, $1 \leq i \leq m$, $1 \leq j \leq k$, such that $\Phi_j(\Gamma_j^i \cup \cdots \cup \Gamma_j^m)$ is a basis for $\Phi_j(H_j^{i-1})$.

By taking appropriate powers (and products) of the elements in Γ_j^i we obtain a set $\Gamma_j'^i$, such that $\{\Phi_j(\gamma) | \gamma \in \Gamma_j'^i\}$ is a complete set of coset representatives of $\Phi_j(H_j^i)$ in

$\Phi_j(H_j^{i-1})$, $1 \leq i \leq m$, $1 \leq j \leq k$. By Proposition 4.5 the set $\Delta^i = \Gamma_1'^i \cdots \Gamma_k'^i$ is a complete set of coset representatives of G_i in G_{i-1}, and the set Δ^i is computable in NC.

Thus, every element $g \in G$ can be written uniquely, as $g = \delta_1 \cdots \delta_m$, where $\delta_i \in \Delta^i$. For any fixed j, $1 \leq j \leq k$,

$$gN_{j-1} = \gamma_1^1 \cdots \gamma_{j-1}^1 \gamma_1^2 \cdots \gamma_{j-1}^2 \cdots \gamma_1^m \cdots \gamma_{j-1}^m N_{j-1},$$

where $\gamma_s^i \in \Gamma_s'^i$ $1 \leq i \leq m$, $1 \leq s \leq j-1$ and $\delta_i H_j^i = \gamma_1^i \cdots \gamma_{j-1}^i H_j^i$.

To compute the δ_i $1 \leq i \leq m$ we sift g through the subspace towers IV.5 defined above for $1 \leq j \leq k$. Each one of the k sifts is in NC, and each one will produce elements $\gamma_j^1 \cdots \gamma_j^m$. Since k has poly(log n) size the entire sifting procedure will be in NC.

We proceed by induction on j, $1 \leq j \leq k$. Let us assume that we have sifted g to acquire the equation,

$$gN_{j-1} = \gamma_1^1 \cdots \gamma_{j-1}^1 \gamma_1^2 \cdots \gamma_{j-1}^2 \cdots \gamma_1^m \cdots \gamma_{j-1}^m N_{j-1}.$$

If we let $\alpha_i = \gamma_1^i \cdots \gamma_{j-1}^i$ for $1 \leq i \leq m$, then $\rho = (\alpha_1 \cdots \alpha_m)^{-1} g \in N_{j-1}$. Since $\Phi_j(N_{j-1})$ is a vector space we can, in parallel, express $\Phi_j(\rho)$ in terms of a basis and pull back the basis elements to elements in N_{j-1}.

For reasons that will become apparent later in the proof we shall not use the basis $\Phi_j(\Gamma_j^1 \cup \cdots \cup \Gamma_j^m)$. Instead, we define $\Upsilon^i = \{\gamma^{(\alpha_{i+1} \cdots \alpha_m)^{-1}} | \gamma \in \Gamma_j^i\}$, and the corresponding set $\Upsilon'^i = \{\gamma^{(\alpha_{i+1} \cdots \alpha_m)^{-1}} | \gamma \in \Gamma_j'^i\}$. Note that $\Phi_j(\Upsilon^i \cup \cdots \cup \Upsilon^m)$ is a basis for $\Phi_j(H_j^i)$.

Using the basis $\Phi_j(\Upsilon^1 \cup \cdots \cup \Upsilon^m)$ for $\Phi_j(N_{j-1})$ we find, in NC, $v_i \in \Upsilon'^i$, such that $\rho N_j = v_1 \cdots v_m N_j$. This implies that $gN_j = \alpha_1 \cdots \alpha_m v_1 \cdots v_m N_j$. Thus we have

$gN_j = \alpha_1\gamma_1 \cdots \alpha_m\gamma_m N_j$, where $\gamma_i \in \Gamma'^i_j$ as desired. □

Corollary 4.8 Given a base B for the solvable group $G \leq Sym(n)$ we can find an NC-efficient SGS for G with respect to B.

Proof: By [4] we can find, in NC, generators for each group in the chain of stabilizers of G with respect to B. □

CHAPTER V

ROUTING ON CAYLEY NETWORKS

In this chapter we construct effective interconnection schemes for multiprocessor networks using Cayley graphs and SGSs. In section 1 we describe a failsoft routing algorithm for these graphs based on the "sifting" procedure described earlier. We also show how normal towers can be used to define Cayley graphs and routing algorithms that perform well, as long as no more than $d-1$ processors fail. In the second section we modify Valiant's permutation routing algorithm to run on these Cayley networks. We conclude this section with several concrete examples. In one of the examples the underlying group is a Sylow-2 subgroup of the symmetric group on n elements (n=power of two). In this case the generators are chosen with great care so that sifting could be applied. Also, techniques from Jerrum [25] are used to reduce the size of the generating set without increasing significantly the routing diameter of the Cayley graph.

Failsoft Routing on SGS Cayley Networks

Let $G = G_0 \leq G_1 \leq \cdots \leq G_k = \{id\}$ be a subgroup tower for $G \leq Sym(n)$. Recall that any SGS $U = \bigcup_{i=1}^{k} U_i$ for the tower has $id \in U_i$, for $1 \leq i \leq k$. We shall assume that all SGSs are effective. By this we mean that sifting, through the SGS, can be performed in $O(kn)$ time. In particular, for $g \in G$ we can compute its unique representation as a product $u_k u_{k-1} \cdots u_1$, $u_i \in U_i$, in $O(kn)$ time. In fact, we can sift $g \in G$ in $O(k^2)$ time by

storing u^{-1} instead of $u \in U$.

Definition 5.1 Let $G \leq Sym(n)$ and let $U = \bigcup_{i=1}^{k} U_i$ be an SGS for G with respect to the tower $G = G_0 \leq G_1 \leq \cdots \leq G_k = \{id\}$. The directed Cayley graph for the SGS is $\Gamma(G, W)$, where $W = U \setminus \{id\}$. The generating set for the undirected Cayley graph, $\Gamma(G, W')$, includes all the inverses of W (i.e., $W' = W \bigcup W^{-1}$).

To compute a route from the vertex g_1 to the vertex g_2 in $\Gamma(G, W)$ we sift $g_1^{-1}g_2$ through the SGS. This gives us the equation,

$$g_1 u_k u_{k-1} \cdots u_1 = g_2,\ u_i \in U_i.$$

From the equation it follows that there is a path from g_1 to g_2 of the form,

$$g_1 \stackrel{u_k}{\rightarrow} g_1 u_k \stackrel{u_{k-1}}{\rightarrow} g_1 u_k u_{k-1} \stackrel{u_{k-2}}{\rightarrow} \cdots \stackrel{u_1}{\rightarrow} g_2. \tag{V.6}$$

Note that some of the u_i, $1 \leq i \leq k$, could be the identity. In this case we ignore the edges $\stackrel{u_i}{\rightarrow}$ (i.e., these edges do not exist in the graph). So the actual length of the path is $k - |I|$, where $u_i = id$ if and only if $i \in I$.

If we sift $g_2^{-1}g_1$ we obtain the equation,

$$g_1 v_1^{-1} v_2^{-1} \cdots v_k^{-1} = g_2,\ v_i \in U_i.$$

For the undirected Cayley graph this gives us a second path from g_1 to g_2,

$$g_1 \stackrel{v_1^{-1}}{\rightarrow} g_1 v_1^{-1} \stackrel{v_2^{-1}}{\rightarrow} g_1 v_1^{-1} v_2^{-1} \stackrel{v_3^{-1}}{\rightarrow} \cdots \stackrel{v_k^{-1}}{\rightarrow} g_2. \tag{V.7}$$

Note that any sequence of edges labels $w_1 w_2 \cdots w_m$ may be interpreted as a path in $\Gamma(G,W)$, starting at vertex $g \in G$ and ending at vertex $gw_1 w_2 \cdots w_m$. Throughout the remainder of this chapter paths are described as sequences of edge labels.

Theorem 5.2 Let $\Gamma(G,W)$ be the undirected Cayley graph constructed from an SGS for the tower $G = G_0 \leq G_1 \leq \cdots \leq G_k = \{id\}$. Then there is an efficient algorithm for computing two disjoint paths of length no more than k between any two vertices in the graph.

Proof: Given nodes g_1 and g_2 in the graph we can use the sift procedure to compute paths (V.6) and (V.7) in $O(nk)$ time (in some cases $O(k^2)$ time). Let P_1 be a prefix of $u_k u_{k-1} \cdots u_1$, $u_i \in U_i$, and let P_2 be a prefix of $v_1^{-1} v_2^{-1} \cdots v_k^{-1}$, $v_i \in U_i$. To prove that paths (V.6) and (V.7) are disjoint it will suffice to prove that if $P_1 = P_2$, then $g_1 P_1 = g_2$.

Let i be the smallest integer such that $u_i \neq id$, and let j be the smallest integer such that $v_j \neq id$. Then i is the smallest integer such that $g_1^{-1} g_2 \in G_{i-1}$, and $g_1^{-1} g_2 \notin G_i$. But $g_1^{-1} g_2 \in G_{i-1}$, and $g_1^{-1} g_2 \notin G_i$ if and only if $g_2^{-1} g_1 \in G_{i-1}$ and $g_2^{-1} g_1 \notin G_i$, and it follows that $i = j$.

Let m,t be minimal such that $u_k u_{k-1} \cdots u_m = v_1^{-1} v_2^{-1} \cdots v_t^{-1}$. We may assume, without loss of generality, that $i \leq m,t \leq k$. Since $v_l = id$ for $1 \leq l < i$ we have $u_k u_{k-1} \cdots u_m = v_i^{-1} v_{i+1}^{-1} \cdots v_t^{-1}$, and $v_i \neq id$ implies that $m = i$. Since $u_l = id$ for $1 \leq l < i$ it follows that $g_1 u_k u_{k-1} \cdots u_m = g_2$. □

The above routing algorithm gives us the ability to route around one faulty processor. We can improve the fault tolerance of the Cayley network by using a SGS that is derived from a normal subgroup tower. In fact, we can create a Cayley graph that has

maximal connectivity.

Theorem 5.3 Let $\Gamma(G,W)$ be the directed Cayley graph constructed from an SGS for the normal tower $G = G_0 \trianglerighteq G_1 \trianglerighteq \cdots \trianglerighteq G_k = \{id\}$. Then there is an efficient algorithm for computing $|W|$ disjoint paths between any two vertices in the graph. Moreover, all of the paths have length no more than $k+1$.

Proof: Because each $G_i \trianglelefteq G$ we may view the U_i as both right and left coset representatives of G_i in G_{i-1}, $1 \leq i \leq k$. Let g_1 (source) and g_2 (destination) be any two nodes in the graph (i.e., $g_1, g_2 \in G$). We shall exhibit $|W|$ node independent paths from g_1 to g_2. Since the graph is symmetric we shall assume, without loss of generality, that $g_1 = id$. This is accomplished by replacing g_1 with the id, and replacing g_2 with $g_1^{-1}g_2$.

If we view the U_i as left coset representatives, then each element $g \in G$ can be written uniquely as $g = u_1 u_2 \cdots u_k$, where $u_i \in U_i$, for $1 \leq i \leq k$. In particular, let $e_1 e_2 \cdots e_k = g_2$ be the result of a sift of g_2.

If $u \in U_i$, then $g_2^{-1} u e_1 e_2 \cdots e_{i-1} \in G_{i-1}$. We use this fact to define the one-to-one function $\psi_i : U_i \rightarrow U_i$. For $u_i \in U_i$ define $\psi_i(u)$ to be the unique element in U_i satisfying the equation,

$$u e_1 e_2 \cdots e_{i-1} G_i \psi_i(u) = g_2 G_i.$$

Define $\gamma_i : U_i \rightarrow G_i$ so that $\gamma_i(u)$ is the unique element in G_i satisfying the equation,

$$u \gamma_i(u) e_1 e_2 \cdots e_{i-1} \psi_i(u) = g_2.$$

Consider the following $|W|$ equations, all of which are equal to g_2,

$$u\gamma_i(u)e_1e_2\cdots e_{i-1}\psi_i(u),\ u \in U_i \setminus \{id\},\ \text{for } 1 \leq i \leq k. \tag{V.8}$$

If we replace $\gamma_i(u)$ with the sift of $\gamma_i(u)$, then we can view the sequence of edges labels in (V.8) as a path from g_1 to g_2 ($g_1 = id$). To complete the proof it will suffice to prove that each path has length at most $k+1$, and that all of the paths are disjoint.

First, let us observe that all of the paths have length no more than $k+1$. Since $\gamma_i(u) \in G_i$ a sift of $\gamma_i(u)$ produces a word in W of length at most $k-i$. Thus, all of these paths described above have length no more than $k+1$.

Let P_1 be a prefix of the path $u\gamma_i(u)e_1e_2\cdots e_{i-1}\psi_i(u)$, and let P_2 be a prefix of the path $\gamma_i(w)e_1e_2\cdots e_{i-1}\psi_i(w)$, where $u, w \in W$. As in the proof of Theorem 5.2 it will suffice to show that $P_1 = P_2$ implies that $P_1 = g_2$.

We start by proving that the two paths $u\gamma_i(u)e_1e_2\cdots e_{i-1}\psi_i(u)$, and $\gamma_i(w)e_1e_2\cdots e_{i-1}\psi_i(w)$, are disjoint if $u, w \in U_i \setminus \{id\}$ and $u \neq w$. First note that it is impossible for a prefix of $u\gamma_i(u)$ to equal a prefix of $w\gamma_i(w)$, because these elements are in different G_i cosets. Let s, t be minimal such that,

$$u\gamma_i(u)e_1e_2\cdots e_s = w\gamma_i(u)e_1e_2\cdots e_t. \tag{V.9}$$

In order for the elements to be in the same G_{i-1} coset it must be the case that $s = t$. But then equation (V.9) implies that $wG_i = uG_i$, and this is a contradiction.

Finally, let us consider two strings of the form, $u\gamma_i(u)e_1e_2\cdots e_{i-1}\psi_i(u)$ and $w\gamma_j(w)e_1e_2\cdots e_{j-1}\psi_j(w)$, where $u \in U_i \setminus \{id\}$, $w \in U_j \setminus \{id\}$, and $i < j$. Because $u\gamma_i(u)$

and $w\gamma_j(w)$ are in different G_i cosets, we can use the same argument as above to prove that no prefix of $u\gamma_i(u)e_1e_2\cdots e_{i-1}$ is equal to a prefix of $w\gamma_j(w)e_1e_2\cdots e_{i-1}$.

Now suppose that there exists $1 \leq t \leq i-1$ and $i \leq s \leq j$ such that $u\gamma_i(u)e_1e_2\cdots e_t = w\gamma_j(w)e_1e_2\cdots e_s$. This implies that $u\gamma_i(u)e_1e_2\cdots e_t \in G_ig_2$, and thus, $u\gamma_i(u)e_1e_2\cdots e_t = g_2$ $(e_{t+1} = \cdots e_{i-1} = \psi_i(u) = id)$. □

Permutation Routing on Cayley Networks

Until now we have only considered the problem of point-to-point routing on Cayley networks. In this section we consider the fundamental problem of permutation routing, also known as the packet routing and the parallel communication. There are two reasons why this problem plays a central role in the design of general purpose multiprocessor networks. First, permutation routing is used as a metric to measure the time needed to distribute information in networks. Second, the solution to this problem is an important ingredient in the simulation of "idealistic" computers (CRCW PRAM) by "realistic" distributed networks.

The following definitions are borrowed from Valiant and Brebner's classic paper on parallel communication [45]. The permutation routing problem is defined as follows. Initially each vertex (processor) possesses a message or package targeted for some vertex in the network. It is assumed that each vertex is the origin of a message and that each vertex is the destination of one message. For practical reasons it is necessary to be able to realize partial permutations; in this case, some subset of the vertex set sends messages and no processor receives more than one message. In fact, the routing scheme we describe solves the more general partial h-relation problem. Initially there are at most h messages

at any node, and no destination receives more than h messages. For all of these problems we would like to route the messages using the fewest number of steps and with as little congestion as possible. A vertex may transmit more than one message at a time, but only one message may travel on an edge in a given time step. Collisions occur when two or more messages wish to traverse the same edge at the same time. When this happens one of the messages is sent, and the other messages are forced to wait on a queue.

An initialized scheme is a pair (Γ, IC), where Γ is a regular directed graph. The initial conditions, IC, specify how the packets and their destinations are distributed at time zero. We assume that each processor sends h messages and receives h messages.

A routing scheme for (Γ, IC) is oblivious if the route of each message depends only on the source and destination, and is not effected by the routes of the other messages. If e is an edge in Γ then traff(e) is the expected number of distinct messages that pass along e. We say that the routing scheme is symmetric if traff(e_1)=traff(e_2) for all edges e_1, e_2 in Γ.

We say that a scheme is nonrepeating if whenever two messages take paths $e_1 e_2 \cdots e_r$ and $e'_1 e'_2 \cdots e'_s$ in which $e_i = e'_j$ and $e_l = e'_m$, then it is the case that $l - i = m - j$ and for all p $(i \leq p \leq l)$ $e_p = e'_{p+j-i}$. In other words, once two routes diverge they remain separated.

Valiant described a simple two-phase routing scheme for the 2-ary m-cube that with high probability runs in $O(m)$ time [44]. A more general proof of the algorithm, together with a permutation routing scheme for shuffle graphs and grid graphs, was given by Valiant and Brebner in [45]. In the first phase of the algorithm messages are sent to random vertices in the graph. In the second phase the messages are routed to their correct destination.

Their proof that the algorithm is successful, with overwhelming probability, relies on the fact that each phase of the algorithm is oblivious, nonrepeating, and symmetric.

We show that Valiant's routing scheme is an effective algorithm for solving the partial h-relation problem on any directed Cayley network constructed from an SGS. To accomplish the first phase of the algorithm we have each message in the network select at random $u_i \in U_i$, for $1 \leq i \leq k$. A message at node s is then sent to node $su_k u_{k-1} \cdots u_1$ along the path described by the edge labels u_i. In the second phase of the algorithm we route each message to its final destination using the point-to-point routing algorithm described in section 1. All the messages execute the above algorithm simultaneously as fast as the queues allow.

Note that the algorithm is oblivious, since the routes described for the messages depend only on their initial position and final destination (and some random information). We also observe that each phase of the algorithm is nonrepeating. To see this, suppose that two messages originating at vertices s and s' follow routes $e_1 e_2 \cdots e_r$ and $e'_1 e'_2 \cdots e'_s$, respectively. If $e_i = e'_j$ and $e_l = e'_m$, then it follows that $\ell(e_i) = \ell(e'_j)$ where $\ell(e)$ denotes the label of edge e (i.e., $\ell((g, gw)) = w$). In fact, we have $\ell(e_i) \cdots \ell(e_l) = \ell(e'_j) \cdots \ell(e'_m)$, and by the uniqueness of the representations, $u_k u_{k-1} \cdots u_1$ $(u_i \in U_i)$, it follows that $k - i = m - j$ and for all p $(i \leq p \leq k)$ $\ell(e_p) = \ell(e'_{p+j-i})$. We may conclude that $e_p = e'_{p+j-i}$, for $i \leq p \leq k$.

If the routing scheme were symmetric we could use Valiant and Brebner's result to prove that the algorithm runs to completion in $O(k)$ time with high probability. Unfortunately, the scheme is not symmetric. However, we can prove a weaker condition that is sufficient to show that the routing scheme is successful with enormous probability.

Let e be an edge in Γ, then traff$(e)=\frac{h}{|U_i|}$ if and only if $\ell(e) \in U_i$ for $1 \leq i \leq k$. To see this, let $e = (g, gu_i)$, and note that any message that passes through e in one phase of the scheme must originate at a vertex g', such that $g = g'u_k u_{k-1} \cdots u_{i+1}$ for $u_j \in U_j$, $i+1 \leq j \leq k$. There are $h(|U_k| \cdots |U_{i+1}|)$ such messages; each has a probability of $(|U_k| \cdots |U_{i+1}|)^{-1}$ of reaching g. Each message that reaches g has a probability of $\frac{1}{|U_i|}$ of traversing e. Thus, traff$(e)=\frac{h}{|U_i|}$ if and only if $\ell(e) \in U_i$ for $1 \leq i \leq k$.

We use this fact together with the following facts, which can be found in [45], to prove our main result.

Fact 5.4 Let $e_1 e_2 \cdots e_s$ be a directed path in the graph Γ. If there are at most m messages that enter and leave this path during a phase of the routing scheme, then for any arbitrary queuing discipline no message is delayed by more than $m-1$.

Fact 5.5 Consider N independent Bernoulli trials each with probability p of success. The probability that at least m of the trials succeed is denoted by $B(m, N, p)$. If we have N independent Poisson trials with respective probabilities $p_1, p_2, \ldots, p_N$ where $\Sigma_{i=1}^{N} p_i = Np$, and if $m \geq Np + 1$ is an integer, then the probability of at least m successes is at most $B(m, N, p)$.

Fact 5.6 If $m \geq Np$ is an integer then,

$$\begin{aligned} B(m, N, p) &\leq (\tfrac{Np}{m})^m (\tfrac{N-Np}{N-m})^{N-m} \\ &\leq (\tfrac{Np}{m})^m e^{N-m} \end{aligned}$$

$(e = 2.71\cdots)$.

Fact 5.4 follows from the observation that the routing scheme is nonrepeating.

Fact 5.5 is a Theorem of Hoeffding [22]. The first inequality in Fact 5.6 is due to Chernoff [13], and the second follows from the inequality $(1+\frac{c}{n})^n < e^c$.

Theorem 5.7 Let $\Gamma(G,W)$ be the directed Cayley graph constructed from an SGS for the tower $G = G_0 \leq G_1 \leq \cdots \leq G_k = \{id\}$ where the coset representatives are U_i for $1 \leq i \leq k$. Let $\alpha = (|U_1| + |U_2| + \cdots + |U_k|)^{-1}$, and let (Γ, IC) be any initialized scheme. Then the probability that some message is delayed by ν or more, in one phase of the permutation routing scheme described above, is

$$(\frac{eh\alpha}{\nu})^{\nu} h|G|,$$

provided that $\nu > h\alpha + 1$.

Proof: We say that a message M intersects some edge e in $\Gamma(G,W)$ in a run of the scheme if its route contains the edge e. Consider some fixed route $R : e_k e_{k-1} \cdots e_1$, where $\ell(e_i) \in U_i$. Let P_M be the probability that message M intersects at least one edge of route R. Let P_{M_i} denote the probability that message M intersects edge e_i. Then,

$$\begin{aligned} \Sigma_M P_M &\leq \Sigma_M P_{M_1} + \Sigma_M P_{M_2} + \cdots + \Sigma_M P_{M_k} \\ &\leq \Sigma_{i=1}^{k} \text{traff}(e_i) \\ &\leq h\alpha. \end{aligned}$$

Now suppose that R is the route taken by some particular message M' (in one phase) for some run of the scheme. Then of the remaining $h|G|-1$ messages the expected number whose paths intersect R is less than $h\alpha$. Thus, we have $h|G|$ independent Poisson trials with probabilities summing to less than $h\alpha$ (we added one dummy trial with probability zero of success). Note that independence is guaranteed by obliviousness. By Fact 5.5 and

5.6 the probability of ν successes is bounded by

$$B(\nu, h|G|, \frac{\alpha}{|G|}) \leq (\frac{eh\alpha}{\nu})^{\nu}, \tag{V.10}$$

provided that $\nu > h\alpha + 1$.

Thus, for any message M', $(\frac{eh\alpha}{\nu})^{\nu}$ is an upper bound for the probability that ν other messages intersect with the route of M'. Fact 5.4 implies that the probability that the message is delayed by ν or more is bounded by $(\frac{eh\alpha}{\nu})^{\nu}$. To finish the proof we multiply this quantity by the number of messages in the initialized scheme. □

Valiant and Brebner point out that this routing scheme can be used to solve the partial h-relation problem. Imagine that we start with an initial scheme in which w of the original $h|G|$ packets are removed. This is equivalent to starting with the initial condition that is specified by a partial h-relation. To prove this more general result we need only modify the proof of Theorem 5.7 so that $w + 1$ dummy Poisson trials are added instead of one. Thus, equation (V.10) still holds for a partial h-relation and the remainder of the proof is unchanged.

In Examples 5.6 and 5.7 Theorem 5.7 is used to examine the effectiveness of the randomized routing scheme on two different Cayley networks. In the first example we construct a Cayley network in which all of the U_i have different sizes. In the second example we consider the other extreme, where all the U_i have the same size.

Example 5.6 Let U be an SGS for the tower $Sym(n) = G^0 \geq G^1 \geq \cdots \geq G^{n-1} = \{id\}$, where $G^i = G_{\{1,2,\cdots,i\}}$. Then $|U_i| = n - i + 1$, $k = n - 1$ and $\alpha = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2}$. If $h = 1$ and $\nu = e(n-1)$, then the probability that a message is delayed by at least $e(n-1)$

(in one phase of the routing scheme) is no more than,

$$(\frac{e\alpha}{\nu})^{\nu}|G| \leq (\frac{\ln n}{\sqrt{n}})^{e(n-1)}.$$

Example 5.7 Let G_h be the automorphism group of a complete binary tree of height h. Label the internal nodes of the tree from top to bottom and from left to right, $v_1, v_2, \ldots, v_{2^h-1}$ (i.e., v_1 is the root of the tree and v_{2^h-1} is the parent of the rightmost leaf). Let $g_i \in G_h$ be the automorphism that flips the subtrees of node v_i, and define $G_i = \langle g_{i+1}, \cdots g_{2^h-1} \rangle$. Let $\Gamma(G_h, W)$ be the directed Cayley graph constructed from an SGS for the subgroup tower, $G_0 > G_1 > \cdots G_{2^h-1} = \{id\}$. Note that $U_i = \{id, g_i\}$ and the graph has $d = k = 2^h - 1$, and $\alpha = k^{\frac{1}{2}}$. If $h = 1$ and $\nu = ek$, then by Theorem 5.7 the probability that a message is delayed by at least ν is no more than,

$$(\frac{e\alpha}{\nu})^{\nu}|G| \leq (\frac{1}{|G|})^{e-1}.$$

If we are only interested in point-to-point routing, then it is possible to create Cayley networks that are more dense than the ones presented above. In fact, in many cases it is possible to reduce the size of the SGS (and hence the degree of the network) by a significant amount and still compute routes (via sift) of a reasonable length. For example, if we replace the SGS in example 5.6 with Jerrum's generators for $Sym(n)$, then we can reduce the degree of the graph by a factor of n and increase the routing diameter by only a factor of 2. The same technique can be used to decrease the degree of the Cayley graph constructed in Example 5.7.

Example 5.8 Consider the tower $Sym(n) = G^0 \geq G^1 \geq \cdots \geq G^{n-1} = \{id\}$, where $G^i = G_{\{1,2,\cdots,i\}}$. Let U_i be a set of coset representatives for G_i in G_{i-1}, and let $W = \{(1,2),(2,3),\ldots,(n,n-1)\}$. For any $u \in U_i$ there exist $w_1, w_2 \in W$ such that $G_i u = G_i w_1 w_2$, $1 \leq i < n-1$. Thus the Cayley graph $\Gamma(Sym(n), W)$ has degree $d = n$ and a routing diameter of $k = 2(n-2)+1$. Note that the routing is done via a sift where elements of U_i are realized as a product of no more than two elements from W.

Example 5.9 As before let G_h be the automorphism group of a complete binary tree of height h and let v_i, g_i and G_i be defined as in Example 5.7. Consider the directed Cayley graph $\Gamma(G_h, W')$, where $W' = \{g_i | i \text{ a power of } 2\}$. In other words, we select the left-most generator from each level of the tree. This reduces the degree of the Cayley network from $2^h - 1$ to h and increases the routing diameter by only a factor of 4. To see this, let $R(h)$ denote the routing diameter of the network $\Gamma(G_h, W')$. It is not difficult to see that $R(0) = 1$ and that $R(h) = 2R(h-1) + 3$ for $h \geq 1$. Solving the recurrence relation we have $R(h) = 2^{h+2} - 3$.

CHAPTER VI

UNIVERSAL BROADCAST

This chapter is an extension of Faber's work on universal broadcast schemes. The first section contains the definitions and methodology used in [18] to obtain an optimal universal broadcast in the d-cube. In the second section we modify several of the definitions, and prove that it is possible to perform a universal broadcast, in optimal time, in a number of Cayley networks. We also answer the question, asked in [18], as to whether or not there exists an optimal universal broadcast scheme for every QCG. In the last section we descibe an optimal universal broadcast in several Cayley networks whose groups are wreath products.

Background and Methodology

A directed Cayley graph $\Gamma(G,W)$ models a multiprocessor network that in one time step may:

(a) use all edges (lines) in parallel

(b) send at most one message per edge

(c) store, retrieve, and operate on data

We say that a processor $g \in G$ broadcasts a message in $\Gamma(G,W)$, if g sends the message to all the other processors in the network.

Definition 6.1 A task graph T is a sequence of edges $\{e_i | i \in I\}$ of a graph Γ, each with an associated direction, labeled by positive integers (called times) $t(e_i)$, satisfying:

(a) $t(e_i) < t(e_j)$ implies that $e_i < e_j$ (see below) or e_i, e_j incomparable

(b) $t(e_i) = t(e_j)$ implies that $i = j$ or e_i, e_j incomparable

Definition 6.2 We say that the edges e_i, e_j satisfy $e_i < e_j$ if and only if there is a directed path in T beginning with e_i and ending with e_j. We say e_i and e_j are incomparable if neither $e_i < e_j$ nor $e_j < e_i$.

The time for a task graph T is $\tau = max_{i \in I} t(e_i)$. Faber uses a task graph to model the broadcast of a message from one processor to the rest of the network. The time step in which the message is passed between two processors is given by the function t.

Definition 6.3 A communication graph C is a collection of task graphs having the property that no edge in C can have a given (time) label more than once.

A universal broadcast is a communication graph consisting of a broadcast from each processor. The time for a communication graph is $\tau(C) = max_{T \in C} \tau(T)$. The requirement that no edge in C is given the same time label more than once guarantees that no collisions occur in message passing.

Each broadcast requires the use of at least $|G| - 1$ edges. Since the total number of edges in the graph is $|G||W|$, it follows that the time needed to complete a universal broadcast is,

$$\tau(C) \geq \lceil \frac{|G|(|G| - 1)}{|G||W|} \rceil = \lceil \frac{(|G| - 1)}{|W|} \rceil.$$

Definition 6.4 Let $S_r(g)$ denote the set of all vertices in the graph a distance of r away from g.

Definition 6.5 A regular ordering of $\Gamma(G,W)$ is an indexing of the vertex set $\{g_0, g_1, \ldots g_{|G|-1}\}$ such that:

(a) $g_0 = \{id\}$

(b) if $g_j \in S_r(g_0)$ and $i \leq j$ then $v_i \in S_r(g_0)$

(c) fix $a \geq 0$ and consider a set of elements in G of the form g_{ad+j}, $1 \leq j \leq d$ then there exists $w_1, w_2, \ldots, w_d \in W$, all different, such that $g_{ad+j} = g_{i_j} w_{s_j}$ with $s_j \leq ad$

The regular ordering is used to define a task graph, T_{id}, with edges (g_{i_j}, g_{ad+j}) as defined above in (c). The time assigned to edge (g_{i_j}, g_{ad+j}) is $a+1$. This task graph defines a broadcast from processor id to the rest of the network.

The broadcast T_{id} is used as a template to define a broadcast T_g, for every $g \in G$. The graph automorphism $A_g : G \rightarrow G$, defined by $A_g(g') = gg'$, is used in conjunction with T_{id} to construct T_g. Let $T_g = A_g(T_{id})$ denote the subgraph of $\Gamma(G,W)$ with vertex set G and edge set $E_{T_g} = \{(gv, gvw) | (v, vw) \in E_{T_{id}}\}$. The function t_g is defined so that $t_g((gv, gvw)) = t_{id}((v, vw))$.

Using condition (c) and the fact that the graph automorphism A_g preserves edge labels (i.e., $t(e) = t(A_g(e))$ and $\ell(e) = \ell(A_g(e))$) Faber proves that $C = \{A_g(T_{id}) | g \in G\}$ is a universal broadcast in $\Gamma(G,W)$.

Lemma 6.6 [Faber] Let T_{id} be a task graph constructed from a regular ordering of $\Gamma(G,W)$ and let $C = \{A_g(T_{id}) | g \in G\}$. Then C describes an optimal universal broadcast in

$\Gamma(G, W)$.

Proof: Each T_g, $g \in G$, defines a broadcast from processor g to the rest of the Cayley network, and each broadcast takes time $\lceil \frac{(|G|-1)}{|W|} \rceil$. To finish the proof we must show that no edge in C is labeled with the same time more than once.

Suppose that there exists an edge e in both T_g and T_h, $g, h \in G$, such that $t_g(e) = t_h(e)$. By the construction of T_g and T_h we know that there exists edges $(v, vw), (v', v'w')$ in T_{id} such that,

$$A_g((v, vw)) = e = A_h((v', v'w')).$$

This implies that $w = \ell(e) = w'$, since A_g and A_h preserve the edge label $\ell(e)$. By condition (c) of the regular ordering and the fact that $t_g(e) = t_h(e)$ it follows that $v_g = v_h$. Thus, $g = h$ and there is only one message passed along edge e at time $t_g(e)$. □

Theorem 6.7 [Faber] The time for a universal broadcast in a d-cube is $\lceil \frac{(2^d-1)}{d} \rceil$.

Proof: A proof of this result may be found in [18]. We give an alternate proof in the next section. □

Universal Broadcast in Cayley Networks

In this section we relax the definition of a regular ordering by leaving out condition (b), and we replace the notion of a task graph with that of a broadcast tree.

Definition 6.8 A regular ordering of $\Gamma(G, W)$ is an indexing of the vertex set $\{g_0, g_1, \ldots g_{|G|-1}\}$ such that,

(a) $g_0 = \{id\}$

(b) for any set of vertices $\{g_{ad+j} | 1 \leq j \leq d, a \geq 0\}$ there exists a set $\{g_{i_j} | 1 \leq j \leq d\}$ with $i_j < ad + 1$ and $g_{i_j} w_j = g_{ad+j}$

Through the remainder of this chapter we shall use the new definition of regular ordering.

Definition 6.9 A broadcast tree for processor g in $\Gamma(G, W)$ is a labeled spanning tree, T_g, that satisfies the following properties. The root of the tree is g and all other nodes have indegree 1. Each edge e in T_g is labeled with a positive integer $t_g(e)$, called time, and for every path starting at g,

$$g \xrightarrow{e_1} g_1 \xrightarrow{e_2} g_2 \xrightarrow{e_3} \cdots \xrightarrow{e_s} g_s,$$

$t_g(e_j) < t_g(e_{j+1})$.

We shall use the technique described in section 1 to find a universal broadcast scheme for several Cayley networks. That is, given $\Gamma(G, W)$ we attempt to find a regular ordering for the network. The regular ordering is used to build a broadcast tree T_{id}, and the broadcast tree is used to define a universal broadcast $C = \{A_g(T_{id}) | g \in G\}$. Recall that the time labels on the edges in T_g are defined by $t_g(A_g(e)) = t_{id}(e)$.

We should point out that it is condition (b) of the regular ordering that allows us to use the automorphisms A_g and the broadcast tree T_{id} to describe an optimal universal broadcast in $\Gamma(G, W)$. The following lemma indicates the significants of condition (b) in the methodology used above.

Lemma 6.10 Let T_{id} be a broadcast tree for processor id in the Cayley network $\Gamma(G, W)$, and let $C = \{A_g(T_{id}) | g \in G\}$ be the communication graph described above. Then C

defines a universal broadcast on Γ if and only if $\ell(e_1) \neq \ell(e_2)$ whenever $t_{id}(e_1) = t_{id}(e_2)$ (e_1, e_2 distinct edges in T_{id}).

Proof: The proof of Lemma 6.6 shows that if $\ell(e_1) \neq \ell(e_2)$ for all edges e_1, e_2 in T_{id} with the same time label, then C is a universal broadcast in Γ.

To prove that this condition is necessary, let us assume that there exist distinct edges e_1, e_2 in T_{id} such that $t_{id}(e_1) = t_{id}(e_2)$ and $\ell(e_1) = \ell(e_2) = w$. Then there exist distinct vertices $g, h \in G$, such that $e_1 = (g, gw)$ and $e_2 = (h, hw)$. The broadcast tree $T_{hg^{-1}}$ contains the edge $e_2 = A_{hg^{-1}}(e_1)$ and $t_{hg^{-1}}(e_2) = t_{id}(e_2)$. This will cause a collision on edge e_2 when the messages broadcast from processors id and hg^{-1} both try to cross e_2 at time $t_{id}(e_2)$. □

Let $W = \{w_1, w_2, \ldots, w_d\}$, $G = \langle W \rangle$ and $\langle w_{d+1} \rangle = Z_q$. Given a regular ordering, $\{id, g_1, \ldots, g_{|G|-1}\}$, for $\Gamma(G, W)$ we shall describe a regular ordering for $\Gamma(H, Y)$, where $H = G \times \langle w_{d+1} \rangle$ and $Y = \{(w, id) | w \in W\} \cup \{(id, w_{d+1})\}$. This will be done by organizing the elements of H into blocks, B_i, of size $d+1$. A typical block will have the form, $B_i = [a_1, a_2, \ldots, a_{d+1}]$, where $a_j \in H \setminus \{id\}$ and there exists $b_j \in \dot{\bigcup}_{l<i} B_l$ with $b_j w_j = a_j$, for $1 \leq j \leq d+1$. The first block is $B_0 = [(w_1, id), (w_2, id), \ldots, (id, w_{d+1})]$.

If we can partition $H \setminus \{id\}$ into $\lceil \frac{|G|q-1}{d+1} \rceil$ blocks (the last block may have less than $d+1$ elements in it), then we have defined a regular ordering for $\Gamma(H, Y)$.

The following notation will be used in our discussion. Let $\rho = \lfloor \frac{|G|-1}{d} \rfloor$ and let $r = (|G| - 1) \bmod d$. Note that $|G| - 1 = \rho d + r$. We denote the element $(g, w_{d+1}^j) \in H$ by g^j, $0 \leq j \leq q-1$.

We describe a procedure, Level(s, m, i, z), that accepts as input the nonnegative integers s, m, i, and z with $0 \leq \rho - s < d$ and $i < q - 1$. The procedure Level will

construct blocks for the processors $g^i_{sd+1}, g^i_{sd+2}, \ldots, g^i_{|G|-1}$ and $g^{i+1}_{md+1}, g^{i+1}_{md+2}, \ldots, g^{i+1}_{nd}$ in $\Gamma(H,Y)$. It is assumed that the processors $(G \times \{id\})w^j_{d+1}$, $0 \le j \le i-1$, $g^i_1, g^i_2, \ldots, g^i_{sd}$ and $g^{i+1}_0, g^{i+1}_1, \ldots, g^{i+1}_{md}$ have already been organized into blocks $B_0, B_1, \ldots, B_{z-1}$.

Procedure Level(s,m,i,z):

(* assume $0 \le \rho - s < d$, $0 \le m$ and $i < q-1$ *)

(1) If $\rho - s = r - 1$, then

(* assume $s > m$ and $n = m+1 \le \rho$ *)

(1.1) For $j := 0$ to $r-2$ do

$$B_{z+j} := [g^i_{(s+j)d+1}, g^i_{(s+j)d+2}, \ldots, g^i_{(s+j)d+d}, g^{i+1}_{md+j+1}]$$

(1.2) $B_{z+r-1} := [g^i_{\rho d+1}, \ldots, g^i_{\rho d+r}, g^{i+1}_{md+r+1}, \ldots, g^{i+1}_{md+d}, g^{i+1}_{md+r}]$

(1.3) $n := m+1$

(2) If $0 \le \rho - s < r-1$, then

(2.1) $x := r - 1 - (\rho - s)$

(* assume $s > m+x$ and $n = m+x+1 \le \rho$ *)

(2.2) For $j := 0$ to $\rho - s - 1$ do

$$B_{z+j} := [g^i_{(s+j)d+1}, g^i_{(s+j)d+2}, \ldots, g^i_{(s+j)d+d}, g^{i+1}_{(m+x)d+j+1}]$$

(2.3) For $j := 0$ to $x-1$ do

$$B_{z+\rho-s+j} := [g^{i+1}_{(m+j)d+1}, \ldots, g^{i+1}_{(m+j)d+d}, g^{i+1}_{(m+x)d+r-x+j}]$$

(2.4) $B_{z+r-1} := [g^i_{\rho d+1}, \ldots, g^i_{\rho d+r}, g^{i+1}_{(m+x)d+r+1}, \ldots, g^{i+1}_{(m+x)d+d}, g^{i+1}_{(m+x)d+r}]$

(2.5) $n := m+x+1$

(3) If $r-1 < \rho - s < d$, then

(3.1) $x := d - (\rho - s - r) + m$

(* assume $s > m$, $s + r > x$ and $n = x + 1 \leq \rho$ *)

(3.2) For $j := 0$ to $r - 1$ do

$$B_{z+j} := [g^i_{(s+j)d+1}, g^i_{(s+j)d+2}, \ldots, g^i_{(s+j)d+d}, g^{i+1}_{md+j+1}]$$

(3.3) For $j := 0$ to $\rho - s - r - 1$ do

$$B_{z+r+j} := 1[g^i_{(s+r+j)d+1}, \ldots, g^i_{(s+r+j)d+d}, g^{i+1}_{xd+j+1}]$$

(3.4) $B_{z+\rho-s} := [g^i_{\rho d+1}, \ldots, g^i_{\rho d+r}, g^{i+1}_{md+r+1}, \ldots, g^{i+1}_{md+d}, g^{i+1}_{xd+\rho-s-r-1}]$

(3.5) For $j := 1$ to $d - (\rho - s - r) - 1$ do

$$B_{z+\rho-s+j} := [g^{i+1}_{(m+j)d+1}, \ldots, g^{i+1}_{(m+j)d+d}, g^{i+1}_{xd+\rho-s-r+j+1}]$$

(3.6) $n := x + 1$

Lemma 6.11 If processors $(G \times \{id\})w^j_{d+1}$, $0 \leq j \leq i - 1$, $g^i_1, g^i_2, \ldots, g^i_{sd}$ and $g^{i+1}_0, g^{i+1}_1, \ldots, g^{i+1}_{md}$ have been organized into blocks $B_0 \ldots B_{z-1}$ and the conditions following lines (1), (2.1) and (3.1) are satisfied, then procedure Level organizes processors $g^i_{sd+1}, g^i_{sd+2}, \ldots, g^i_{|G|-1}$ and $g^{i+1}_{md+1}, g^{i+1}_{md+2}, \ldots, g^{i+1}_{nd}$ into blocks.

Proof: We must show that for each new block $B_i = [a_1, \ldots a_{d+1}]$ defined by procedure Level there exists $b_j \in \dot{\bigcup}_{l<i} B_l$ with $b_j w_j = a_j$, $1 \leq j \leq d + 1$. The first d elements in each block are organized with respect to the regular ordering given for $\Gamma(G, W)$. Thus, it will suffice to show that there exists $b \in \dot{\bigcup}_{l<i} B_l$ with $bw_{d+1} = a_{d+1}$. One checks that the bounds on m and x given after lines (1), (2.1) and (3.1) insure that this requirement is met. Note that $n \leq \rho$ guarantees that the indices are defined (i.e., not too large). □

Lemma 6.12 Let $W = \{w_1, w_2, \ldots, w_d\}$, $G = \langle W \rangle$ and $\langle w_{d+1} \rangle = Z_2$. If $\{id, g_1, \ldots, g_{|G|-1}\}$ is a regular ordering for $\Gamma(G, W)$ and $\rho - 1 > d \geq 2$, then we can construct a regular ordering for $\Gamma(H, Y)$, where $H = G \times \langle w_{d+1} \rangle$ and $Y = \{(w, id) | w \in W\} \cup \{(id, w_{d+1})\}$.

Proof: Let $m = \lfloor \frac{\rho-1}{d} \rfloor$ and let $s = md + 1$. The first s blocks are $B_j = [g^0_{jd+1}, \ldots, g^0_{jd+d}, g^1_j]$, $0 \leq j \leq md$. At this point we call Level$(s, m, 0, md + 1)$.

Recall that the following preconditions must be satisfied for procedure Level to operate correctly:

(a) processors $g^0_1, g^0_2, \ldots, g^0_{sd}$ and $g^1_0, g^1_1, \ldots, g^1_{md}$ are organized into blocks

(b) $0 \leq \rho - s < d$

(c) if $\rho - s = r - 1$ then $s > m$ and $n = m + 1 \leq \rho$

(d) if $0 \leq \rho - s < r - 1$ then $m + r - 1 < \rho$ and $m + r + s \leq 2\rho$

(e) if $r - 1 < \rho - s < d$ then $s > m$, $d + m < \rho$ and $d + r + s + m + 1 \leq 2\rho$

Condition (a) is satisfied by blocks $B_0, \ldots, B_{md}$, and condition (b) follows from the definitions of s and m. To prove that conditions (c), (d) and (e) hold it suffices to show that $\rho > d+m$. Note that $\rho(d-1) > d^2-1$, since $\rho > d+1$ (hypothesis). Thus, $\rho > d + \frac{\rho-1}{d}$ and the result follows. When procedure Level is finished we will have organized into blocks the elements $g_1, g_2, \ldots, g_{|G|-1}$ and $g^1_0, g^1_1, \ldots, g^1_{nd}$.

There are $|G| - (nd + 1)$ elements left in $H \setminus \{id\}$ that have not been processed into blocks. Let $p = \lfloor \frac{|G|-(nd+1)}{d+1} \rfloor$, let $z = \frac{|G|+nd}{d+1}$ and let $y = (|G| - (nd + 1)) \bmod d + 1$. We build p more (full) blocks $B_{z+j} = [g^1_{(n+j)d+1}, \ldots, g^1_{(n+j)d+d}, g^1_{|G|-1-j}]$, where $0 \leq j \leq p - 1$. If $y \neq 0$ the last block is $B_{z+p} = [g^1_{(n+p)d+1}, \ldots, g^1_{(n+p)d+y}, \epsilon, \ldots, \epsilon]$. □

In the proof of Lemma 6.12 the generators $w_1, \ldots, w_d$ are used to process the elements in the coset $G \times \{id\}$. The procedure Level is used to prepare the coset $(G \times \{id\})w_{d+1}$ for processing by the generators $w_1, \ldots, w_d$. The next procedure uses this strategy to find a regular ordering when there are more than two cosets (i.e., $\langle w_{d+1} \rangle = Z_q$

and $q \geq 3$).

Procedure Process(ρ, d, q):

(1) $n := 0$; $i := 0$; $z_i := 0$

(2) While $i < q - 1$ do

(2.1) $B_{z_i} = [g^i_{nd+1}, \ldots, g^i_{nd+d}, g_0^{i+1}]$

(2.2) $m := \lfloor \frac{\rho-(n+1)}{d} \rfloor$

(2.3) $s := md + n + 1$

(2.4) For $j := 1$ to md do

$B_{z_i+j} = [g^i_{(n+j)d+1}, \ldots, g^i_{(n+j)d+d}, g_j^{i+1}]$

(2.5) Level$(s, m, i, z_i + md + 1)$

(2.6) $i := i + 1$

(2.7) $n :=$value returned from Level

(2.8) $z_i := \frac{i|G|+nd}{d+1}$

(3) $p = \lfloor \frac{|G|-(nd+1)}{d+1} \rfloor$

(4) $z_{q-1} = \frac{(q-1)|G|+nd}{d+1}$

(5) $y = (|G| - (nd + 1)) \bmod d + 1$

(6) For $j := 0$ to $p - 1$ do

(6.1) $B_{z_{q-1}+j} = [g^{q-1}_{(n+j)d+1}, \ldots, g^{q-1}_{(n+j)d+d}, g^{q-1}_{|G|-1-j}]$

(7) $B_{z_{q-1}+p}[g^{q-1}_{(n+p)d+1}, \ldots, g^{q-1}_{(n+p)d+y}, \epsilon, \ldots, \epsilon]$

Lemma 6.13 Let $W = \{w_1, w_2, \ldots, w_d\}$, $G = \langle W \rangle$ and $\langle w_{d+1} \rangle = Z_q$. If we have a regular ordering, $\{id, g_1, \ldots, g_{|G|-1}\}$, for $\Gamma(G, W)$, $2 \leq d < \rho - 1$ and $3 \leq q$, then we can construct a regular ordering for $\Gamma(H, Y)$, where $H = G \times \langle w_{d+1} \rangle$ and $Y = \{(w, id) | w \in W\} \cup$

$\{(id, w_{d+1})\}$.

Proof: The proof is analogous to the proof of Lemma 6.12. One checks that before each call to procedure Level the following preconditions are satisfied:

(a) elements $(G \times \{id\})w_{d+1}^j$, $0 \leq j \leq i-1$, $g_1^i, g_2^i, \ldots, g_{sd}^i$ and $g_0^{i+1}, g_1^{i+1}, \ldots, g_{md}^{i+1}$ are organized into blocks

(b) $0 \leq \rho - s < d$

(c) if $\rho - s = r - 1$ then $s > m$ and $n = m + 1 \leq \rho$

(d) if $0 \leq \rho - s < r - 1$ then $m + r - 1 < \rho$ and $m + r + s \leq 2\rho$

(e) if $r - 1 < \rho - s < d$ then $s > m$, $d + m < \rho$ and $d + r + s + m + 1 \leq 2\rho$ □

Lemma 6.14 Let $w_1 = (\alpha, id)$ and $w_2 = (id, \beta)$ be generators for $H = Z_{q'} \times Z_q$. There is a regular ordering for the Cayley network $\Gamma(H, \{w_1, w_2\})$.

Proof: Observe that there is a unique regular ordering for $\Gamma(\langle w_1 \rangle, \{w_1\})$. If we make two slight modifications to procedure Process, then the procedure can be used to define a regular ordering for $\Gamma(H, \{w_1, w_2\})$. First, we leave out line (2.4) (i.e., skip the call to procedure Level). Second, we replace line (2.7) with "n:=m". The procedure call, Process$(q' - 1, 1, q)$, will result in a regular ordering of $\Gamma(H, \{w_1, w_2\})$. □

Lemma 6.15 Let $L = \langle w_1, \ldots, w_d, w_{d+1} \rangle$ be an abelian group. Let $G = \langle w_1, \ldots, w_d \rangle$ and let $L/G = Z_q$, $q \geq 2$. Let T_1 be a broadcast tree for processor id in $\Gamma(H, Y)$, where $H = G \times \langle \alpha \rangle$, $Y = \{(w, id) | w \in \{w_1, \ldots, w_d\}\} \cup \{(id, \alpha)\}$ and α is a generator for Z_q. Let T_2 be the broadcast tree obtained by replacing, edges of T_1 labeled α with edges labeled w_{d+1} and nodes labeled (g, α^i) with gw_{d+1}^i, $0 \leq i < q$. Then T_2 is a broadcast tree, in $\Gamma(L, \{w_1, \ldots, w_{d+1}\})$, for processor id.

Proof: We need only check that all of the nodes in T_2 are distinct. If $gw_{d+1}^i = g'w_{d+1}^j$, then $i = j$ since q is the smallest integer for which $w_{d+1}^q \in G$. Thus $g = g'$ and we are done. □

Theorem 6.16 Let $G = \langle w_1, w_2, \ldots, w_d \rangle$ be an abelian group, and let $G_i = \langle w_1, w_2, \ldots, w_i \rangle$. If $G_i < G_{i+1}$ for $1 \leq i < d$, then we can find a regular ordering for $\Gamma(G, W)$.

Proof: Using Lemmas 6.12-6.15 we can extend a regular ordering for $\Gamma(G_i, W_i)$ to a regular ordering for $\Gamma(G_{i+1}, W_{i+1})$. The only cases not covered by the Lemmas are:

Case 1 $i = d = 2$ and $|G_i| < 9$,

Case 2 $i = d = 3$ and $|G_i| < 16$,

Case 3 $i = d = 4$ and $|G_i| < 25$,

Case 4 $i = d = 5$ and $|G_i| < 36$.

Using Lemma 6.15 it suffices to check that there is a regular ordering for the following Cayley networks:

(a) $\Gamma(G_1 = Z_2 \times Z_2 \times Z_q, \{w_{11}, w_{12}, w_{13}\})$

(b) $\Gamma(G_2 = Z_2 \times Z_3 \times Z_q, \{w_{21}, w_{22}, w_{23}\})$

(c) $\Gamma(G_3 = Z_2 \times Z_4 \times Z_q, \{w_{31}, w_{32}, w_{33}\})$

(d) $\Gamma(G_4 = Z_2 \times Z_2 \times Z_2 \times Z_q, \{w_{41}, w_{42}, w_{43}, w_{44}\})$

(e) $\Gamma(G_5 = Z_2 \times Z_2 \times Z_3 \times Z_q, \{w_{51}, w_{52}, w_{53}, w_{54}\})$

(f) $\Gamma(G_6 = Z_2 \times Z_2 \times Z_2 \times Z_2 \times Z_q, \{w_{61}, w_{62}, w_{63}, w_{64}, w_{65}\})$

(g) $\Gamma(G_7 = Z_2 \times Z_2 \times Z_2 \times Z_3 \times Z_q, \{w_{71}, w_{72}, w_{73}, w_{74}, w_{75}\})$

(h) $\Gamma(G_8 = Z_2 \times Z_2 \times Z_2 \times Z_2 \times Z_2 \times Z_q, \{w_{81}, w_{82}, w_{83}, w_{84}, w_{85}, w_{86}\})$

The set $\{w_{i1}, \ldots, w_{ir_i}\}$, $1 \leq i \leq 8$, is the canonical set of generators for the group G_i. □

Corollary 6.17 The time for a universal broadcast in a d-cube is $\lceil \frac{(2^d-1)}{d} \rceil$.

Corollary 6.18 Let G be a finite abelian group with $G = \langle W \rangle$. If we can order the elements of W so that $w_i \notin \langle w_1, w_2, \ldots w_{i-1} \rangle$, then we can find a regular ordering for $\Gamma(G, W)$.

Note that in Theorem 6.16 we required that the generating set for G satisfy the property that $w_{i+1} \notin G_i$. The next example shows that without this condition one can construct Cayley networks that cannot be regularly ordered. This answers the question posed by Faber as to whether or not every QCG can be regularly ordered.

Example 6.10 Let $G = Z_{2n}$ and let $W = \{1, n, n+1\}$, then the Cayley network $\Gamma(G, W)$ has diameter $n-1$. Since a broadcast will need at least $n-1$ time steps, there is no regular ordering of the Cayley network when $n \geq 5$.

We conclude this section with a result that shows that an optimal universal broadcast exists for any Cayley network, $\Gamma(G, W)$, where G is a cyclic group and each $w \in W$ is a generator for G.

We say that a path

$$g \xrightarrow{e_1} g_1 \xrightarrow{e_2} g_2 \xrightarrow{e_3} \cdots \xrightarrow{e_s} g_s,$$

in $\Gamma(G, W)$ is a w-path if $\ell(e_i) = w$ for $1 \leq i \leq s$.

Lemma 6.19 Let $\Gamma(G, W)$ be a Cayley network where G is a cyclic group and each $w \in W$ is a generator for G. If $G = X \dot{\cup} Y$, $\{r_1, r_2, \ldots, r_m\} = R \subseteq W$ and $m \leq |Y|$, then there exists $x_1, \ldots, x_m \in X$ and distinct $y_1, \ldots, y_m \in Y$, such that $x_i r_i = y_i$ for $1 \leq i \leq m$.

Proof: We give a constructive proof that uses induction on $m = |R|$ to find the x_i and y_i for $1 \leq i \leq m$. Since each $w \in W$ is a generator, the statement holds for $|R| = 1$. Suppose the lemma is true for $|R| \leq m$ and we have found $x_1, \ldots, x_m \in X$ and distinct $y_1, \ldots, y_m \in Y$, such that $x_i r_i = y_i$ for $1 \leq i \leq m$. Let $\{y_1, \ldots, y_m\} = Y'$ and let

$$x \xrightarrow{e_1} y_{i_1} \xrightarrow{e_2} y_{i_2} \xrightarrow{e_3} \cdots \xrightarrow{e_{s-1}} y_{i_{s-1}} \xrightarrow{e_s} y$$

be a r_0-path from X to $Y \setminus Y'$, where $r_0 \in W \setminus \{r_1, \ldots, r_m\}$. If this path has length 1, then we are done. If the path has length $s > 1$, then we replace the path with an r_0-path from X to $Y \setminus Y'$ of length at most $s - 1$.

The procedure we describe for computing the new r_0-path has the property that after the k^{th} step either an r_0-path of length at most $s - 1$ is found or the following conditions are true:

(1) $x_i r_i = y_i$, $1 \leq i \leq m$,

(2) $y_{i+1} r_i = y$, $0 \leq i \leq k$,

(3) $x_i r_{i-1} = y_{i+1}$, $1 \leq i \leq k$.

We assume, without loss of generality, that all of the y_{i_j}, $1 \leq j \leq s - 1$, are distinct elements in Y' and that $y_{s_{i-1}} = y_1$. Thus, $y_1 r_0 = y$ and the three conditions are true for $k = 0$. Assuming that the conditions hold for $k = j - 1$ we outline the j^{th} step of the procedure.

Case 1: (j odd) If $x_j r_{j-1} = z \in Y \setminus Y'$, then replace

$$\begin{aligned} x_{j-1}r_{j-1} = y_{j-1} \quad &\text{with} \quad x_j r_{j-1} = z, \\ x_{j-3}r_{j-3} = y_{j-3} \quad &\text{with} \quad x_{j-2}r_{j-3} = y_{j-1}, \\ \vdots \qquad &\qquad \vdots \\ x_2 r_2 = y_2 \quad &\text{with} \quad x_3 r_2 = y_4. \end{aligned}$$

This leaves the element y_2 unused. Now we move y_2 out of Y', and we have the r_0-path $x_1 r_0 = y_2$ of length 1 from X to $Y \setminus Y'$.

If $x_j r_{j-1} = x \in X$, then $x r_j = y$ and we replace

$$\begin{aligned} x_j r_j = y_j \quad &\text{with} \quad x r_j = y, \\ x_{j-2}r_{j-2} = y_{j-2} \quad &\text{with} \quad x_{j-1}r_{j-2} = y_j, \\ \vdots \qquad &\qquad \vdots \\ x_1 r_1 = y_1 \quad &\text{with} \quad x_2 r_1 = y_3. \end{aligned}$$

This leaves y_1 free and we can move y_1 out of Y'. Observe that there is an r_0-path from X to $Y \setminus Y'$ of length $s-1$.

Case 2: (j even) If $x_j r_{j-1} = z \in Y \setminus Y'$, then we replace

$$\begin{aligned} x_{j-1}r_{j-1} = y_{j-1} \quad &\text{with} \quad x_j r_{j-1} = z, \\ x_{j-3}r_{j-3} = y_{j-3} \quad &\text{with} \quad x_{j-2}r_{j-3} = y_{j-1}, \\ \vdots \qquad &\qquad \vdots \\ x_1 r_1 = y_1 \quad &\text{with} \quad x_2 r_1 = y_3. \end{aligned}$$

We can move y_1 out of Y' resulting in an r_0-path of length $s-1$.

If $x_j r_{j-1} = x \in X$ then we have $x r_j = y$ and we replace

$$\begin{array}{llll} x_j r_j = y_j & \text{with} & x r_j = y, \\ x_{j-2} r_{j-2} = y_{j-2} & \text{with} & x_{j-1} r_{j-2} = y_j, \\ \vdots & & \vdots \\ x_2 r_2 = y_2 & \text{with} & x_3 r_2 = y_4. \end{array}$$

Hence y_2 is unused and we have the r_0-path $x_1 r_0 = y_2$ from X to $Y \setminus Y'$.

Otherwise $x_j r_{j-1} \in Y' \setminus \{y_1, \ldots y_j\}$, and we may order the elements, with indices greater than j, so that $x_j r_{j-1} = y_{j+1}$.

To finish the proof observe that for $j = m$ condition (2) guarantees that $x_m r_{m-1} \notin Y'$. □

Corollary 6.20 Let G be a finite cyclic group and let $W \subseteq G$ such that $G = \langle w \rangle$, for all $w \in W$. Then we can find a regular ordering for $\Gamma(G, W)$. Moreover, any separating set for the graph must have size at least $|W|$.

Universal Broadcast Schemes and Wreath Products

Let $G \leq Sym(A)$ generated by $W = \{w_1, \ldots, w_d\}$, and let $A \dot{\cup} A_1$ denote the disjoint union of two copies of A. Let α be the permutation that interchanges each $a \in A$ with it counterpart in A_1. We extend each $w \in W$ to a permutation on $A \dot{\cup} A_1$, such that w acts trivially on A_1. Then the wreath of G by Z_2, $G \wr Z_2$, is the subgroup of $Sym(A \dot{\cup} A_1)$ generated by $W \cup \{\alpha\}$.

Given a regular ordering for $\Gamma(G, W)$, we have shown how to construct a regular ordering for $\Gamma(H, Y)$, where $H = G \times \langle w' \rangle$ and $Y = \{(w, id) | w \in W\} \cup \{(id, w')\})$. In this section we describe a process for constructing a regular ordering for $\Gamma(G \wr Z_2, W \cup \{\alpha\})$. The following facts about $G \wr Z_2$ are needed:

(a) $|G \wr Z_2| = |G|^2 2$

(b) $G \times G \lhd G \wr Z_2$

(c) $G \wr Z_2 = G \times G \dot{\cup} \alpha(G \times G)$

Thus, each element in $G \wr Z_2$ can be written uniquely as a product $\alpha^e(g_1, g_2)$, where $e \in \{0,1\}$ and $(g_1, g_2) \in G \times G$. For further information on wreath products we refer the reader to [36].

Our earlier results relied on the fact that the generator, (id, w'), commuted with the generators $\{(w, id) | w \in W\}$. The generator α does not have this property. We have, instead, the following equation,

$$\alpha(g_1, g_2) = (g_2, g_1)\alpha. \tag{VI.11}$$

Let $\{g_0, g_1, \ldots, g_{|G|-1}\}$ be a regular ordering for $\Gamma(G, W)$, and recall that $\rho = \lfloor \frac{|G|-1}{d} \rfloor$ and $r = (|G| - 1) \bmod d$. We will use the regular ordering for $\Gamma(G, W)$ and the generators $w_1, \ldots, w_d$ to process the cosets $G \times \{id\}$ of $G \wr Z_2$. Hence, for some fixed $g \in G$ the $w_1, \ldots, w_d$ will access, in a single step, elements $\alpha^e(g_{ad+1}, g), \ldots, \alpha^e(g_{ad+d}, g)$, $0 \leq a \leq \rho - 1$ and $e \in \{0,1\}$. We denote this block of d elements by $\alpha^e(D_a, g)$ (note that D_a is a block from the regular ordering of G).

The generator α will process the elements $\alpha^e(id, g)$, $g \in G$ and $e \in \{0,1\}$. The last r elements in each coset $G \times \{id\}$ of $G \wr Z_2$, except for the cosets with representatives (id, id), $\alpha(id, id)$, $(id, g_{\rho d+1})$, $(id, g_{\rho d+2}), \ldots, (id, g_{\rho d+r})$, are also processed by α.

As before, the major obstacle is the ordering of the last r elements in each coset $G \times \{id\}$ of $G \wr Z_2$. Procedure Level handled this problem for the regular ordering of

$\Gamma(G \times Z_q, Y)$. For the Cayley network $\Gamma(G \wr Z_2, \{w_1, \ldots, w_d, \alpha\})$ the sets R and R' are used to manage these elements. For $g, g' \in G$ let $(R, g) = \{(g_{\rho d+1}, g), \ldots, (g_{\rho d+r}, g)\}$ and let $R' = \{\alpha(g_{r+1}, g'), \ldots, \alpha(g_{r+1}, g')\}$.

The procedures Access1 and Access2 describe the order in which the generators $w_1, \ldots, w_d$ and α process the elements of $G \wr Z_2$. In both procedures, $r' = 0$ if $r = 0$ and $r' = 1$ if $r > 0$.

Procedure Access1:

(*The order in which $w_1, \ldots, w_d$ process elements of $G \wr Z_2$.*)

Step 1

(*process the blocks of $G \times \{id\}$*)

(1.1) For $i = 0$ to $\rho - 1$ do

(D_i, id)

Step 2 If $r' = 1$ then

(*complete the coset $G \times \{id\}$*)

(2.1) (R, id)

(2.2) $\alpha(R', g_1)$

Step 3

(*process the blocks of $\alpha(G \times \{id\})$*)

(3.1) For $i = 0$ to $\rho - 1$ do

$\alpha(D_i, id)$

Step 4 If $r' = 1$ then

(*complete the coset $\alpha(G \times \{id\})$*)

(4.1) $\alpha(R, id)$

(4.2) $\alpha(R', g_2)$

Step 5

(*allow generator α to finish $\{id\} \times G$ and $\alpha(\{id\} \times G)$*)

(5.1) For $j := 1$ to ρd do

(D_0, g_j)

(5.2) For $j := 1$ to $|G| + r - 2\rho - 2r'$

(D_1, g_j)

Step 6

(*process cosets $(id, g_{\rho d+1})(G \times \{id\}), \ldots, (id, g_{\rho d+r})(G \times \{id\})$*)

(6.1) For $j := 1$ to r do

(6.1.1) For $i = 0$ to $\rho - 1$ do

$(D_i, g_{\rho d+j})$

(6.1.2) $(R, g_{\rho d+j})$

(6.1.3) $\alpha(R', g_{j+2r'})$

Step 7

(*process cosets $\alpha(id, g_{\rho d+1})(G \times \{id\}), \ldots, \alpha(id, g_{\rho d+r})(G \times \{id\})$*)

(7.1) For $j := 1$ to r do

(7.1.1) For $i = 0$ to $\rho - 1$ do

$\alpha(D_i, g_{\rho d+j})$

Step 8

(*process remaining blocks in $G \times G$*)

(8.1) For $j := |G| + r - 2\rho - 2r' + 1$ to ρd do

(D_1, g_j)

(8.2) For $i = 2$ to $\rho - 1$ do

(8.2.1) For $j := 1$ to ρd do

(D_i, g_j)

Step 9

(*process remaining blocks in $\alpha(G \times G)$*)

(9.1) For $j := \rho d$ down to $r + 2r' + 1$ do

(9.1.1) For $i := 0$ to $\rho - 1$ do

$\alpha(D_i, g_j)$

(9.2) For $j := r + 2r'$ down to 1 do

(9.2.1) For $i = 1$ to $\rho - 1$ do

$\alpha(D_i, g_j)$

Procedure Access2:

(*The order in which α processes the elements of $G \wr Z_2$.*)

Step A

(*process elements in $\alpha(\{id\} \times G)$*)

(*coincides with step 1, step 2 and one move of step 3 $[i := 0]$*)

(A.1) For $j = 0$ to $\rho + r'$ do

$\alpha(id, g_j)$

Step B

(*process elements in $\{id\} \times G$*)

(*coincides with step 3 [$i := 1$ to $\rho - 1$] and step 4*)

(B.1) For $j = 1$ to $\rho - 1 + r'$ do

(id, g_j)

Step C

(*process the remaining elements in $\{id\} \times G$ and $\alpha(\{id\} \times G)$*)

(*coincides with step 5*)

(C.1) For $j = \rho + r'$ to $|G| - 1$ do

(id, g_j)

(C.2) For $j = \rho + r' + 1$ to $|G| - 1$ do

$\alpha(id, g_j)$

Step D If $r' = 1$ then

(*complete unfinished part of R' blocks started in steps 2, 4 and 6*)

(*coincides with step 6*)

(D.1) For $j = 1$ to $r + 2$ do

(D.1.1) For $i := 1$ to r do

$\alpha(g_i, g_j)$

Step E If $r' = 1$ then

(*process the last r elements in cosets $\alpha(id, g)(G \times \{id\})$*)

(*step 6 has started by this time*)

(E.1) For $j = 1$ to $\rho d + r$ do

(E.1.1) For $i := \rho d + 1$ to $\rho d + r$ do

$\alpha(g_i, g_j)$

Step F If $r' = 1$

(*process the last r elements in cosets $(id, g)(G \times \{id\})$*)

(*step 7 has started by this time *)

(F.1) For $j = 1$ to ρd do

(F.1.1) For $i := \rho d + 1$ to $\rho d + r$ do

(g_i, g_j)

Step G

(*process cosets $\alpha(id, g)(G \times \{id\})$ in the opposite order of step 9*)

(*step 8 has started by this time*)

(G.1) For $j = 1$ to $r + 2r'$ do

(G.1.1) For $i := \rho d$ down to $d + 1$ do

$\alpha(g_i, g_j)$

(G.2) For $j = r + 2r' + 1$ to ρd do

(G.2.1) For $i := \rho d$ down to 1 do

$\alpha(g_i, g_j)$

Theorem 6.21 Let $G \leq Sym(A)$ generated by $W = \{w_1, \ldots, w_d\}$, and let $G \wr Z_2 = \langle w_1, \ldots, w_d, \alpha \rangle$, α defined as above. If we have a regular ordering, $\{id, g_1, \ldots, g_{|G|-1}\}$, for $\Gamma(G, W)$, and

(a) $\rho \geq r + 1 + \frac{d - r(r+2)}{|G|}$

(b) $|G| + r \geq 2\rho + 2r'$

(c) $\rho \geq 2$

then we can construct a regular ordering for $\Gamma(G \wr Z_2, \{w_1, \ldots, w_d, \alpha\})$.

Proof: Procedures Access1 and Access2 describe a regular ordering of $G \wr Z_2$ if the elements accessed at any given time step can be reached, by the generators, from elements that have already been processed. If this is true, then condition (b) of definition 6.8 is satisfied.

First observe that by the time step 5 is completed all of the elements in $\{id\} \times G$ and $\alpha(\{id\} \times G)$ have been processed by α. Thus, the only steps in Access1 that we need worry about are (2.2), (4.2), and (5.1). Steps (2.2) and (4.2) are not a problem, since $\rho \geq 2$. Since step B in procedure Access2 is completed before step (5.1) is started, it follows that the elements defined in this step are part of a regular ordering.

Equation (VI.11) implies that the generator α can access the element $\alpha^e(g_i, g_j) \in G \wr Z_2$ if and only if the element $\alpha^{e+1}(g_j, g_i)$ has been processed in a previous step.

All the elements processed in step A meet this criterion. The elements processed by α in Step B are part of a regular ordering, since step B is not started until the first move of step 3 is finished. Step C presents no problem, since it is not started until after the cosets $G \times \{id\}$ and $\alpha(G \times \{id\})$ are processed.

Condition (b) gaurantees step (5.1) has finished, before step D is started. Step E is not a problem, since step 6 starts processing elements by the time it begins. Likewise step 7 has started processing elements by the time step F is started.

Condition (a) gaurantees that step F runs to completion. In step G we have forced α to process the unordered elements of $\alpha(G \times G)$ in the opposite order that the generators $w_1, \ldots, w_d$ are working. Thus, at some point in time the two procedures will converge on a set of elements of $G \wr Z_2$ of size at most d. These can then be handled by a subset of the generators $w_1, \ldots, w_d$. □

Corollary 6.22 We can construct a regular ordering for $\Gamma(Z_q \wr Z_2, \{w_1, \alpha\})$, where w_1 is a generator for Z_q and $q \geq 2$.

Proof: Note that $\rho = q - 1$, $r = 0$ and $d = 1$. If $q \geq 3$, then $\rho \geq 2$ and generator w_1 can follow the steps outlined in procedure Access1. Every element processed by w_1, after the first step, is multiplied by the generator α. The case $q = 2$ is a simple exercise. □

Corollary 6.23 We can construct a regular ordering for $\Gamma(l^k Z_2, \{\alpha_1, \ldots, \alpha_k\})$, where the α_i, $1 \leq i \leq k$, are the canonical generators defined above.

Proof: We use Corollary 6.23 to define a regular ordering for $\Gamma(Z_2 \wr Z_2, \{\alpha_1, \alpha_2\})$. Then given a regular ordering for $\Gamma(l^i Z_2, \{\alpha_1, \ldots, \alpha_i\})$, $i \geq 2$, Theorem 6.21 is used to define a regular ordering for $\Gamma(l^{i+1} Z_2, \{\alpha_1, \ldots, \alpha_{i+1}\})$. □

CHAPTER VII

AN ALGEBRAIC ANALYSIS OF THE MOEBIUS GRAPH

Leland and Solomon introduced a family of trivalent graphs, called Moebius graphs, which had a diameter that was 25% smaller than any family of trivalent graphs previously defined. Because of its small diameter and degree the Moebius graph was proposed as an effective interconnection scheme for multiprocessor networks [30]. The authors suggested, for future research, two open problems concerning the Moebius graph. First, find an efficient algorithm for computing optimal (shortest) paths for the Moebius graph. Second, determine the exact diameter of the Moebius graph.

In this chapter we use algebraic techniques to analyze the Moebius graph, and solve the two problems stated above. We reduce the graph theory problem of routing, to the algebraic problem of computing generating sequences. This reduction follows from the observation that the Moebius graph is isomorphic to a quotient Cayley graph (QCG).

In the past ad hoc methods have been used to show that specific nonsymmetric networks, such as the de Bruijn and Shuffle-exchange, can be viewed as QCGs. We state a more general result which says that a graph is isomorphic to a QGC if and only if it satisfies the "labeling" property. Given a "labeling" for the graph, this result not only identifies the graph as an QCG, but it describes the groups needed to define the QCG and an isomorphism between the graphs. We use these groups to guide the development

of our routing algorithm and to prove that the algorithm is correct.

In the first section we present background information about the Moebius graph and prove that the graph is isomorphic to to an QCG. We also introduce the notation and the algebraic tools needed in later sections. The second section contains a description of the optimal routing algorithm and a proof of correctness. In the last section we determine the exact diameter of the Moebius graph.

The Moebius Graph and Algebraic Tools

Notation:

- Let $(Z_2)^n = \{(x_{n-1}, \ldots, x_1, x_0) | x_i \in \{0,1\}\}$
- Let $\vec{0} = (0,0,\ldots,0)$ and let $\vec{1} = (1,1,\ldots,1)$
- For $v \in (Z_2)^n$, let $wt(v) = \Sigma_{i=0}^{n-1} v_i$
- For $v \in (Z_2)^n$, let $pr(v) = wt(v) \bmod 2$
- Let $J \subseteq \{0,2,\ldots,n-1\}$, then we call $v \in (Z_2)^n$ the characteristic vector for J if $v_j = 1$ if and only if $j \in J$

Definition 7.1 For any integer $n \geq 2$ let $M_n(V,E)$ denote the Moebius graph of order n. The graph has $V = (Z_2)^n$, and $E = \{(v, v^s) | v \in V \text{ and } s \in \{\rho, \rho^{-1}, \delta\}\}$, where ρ and δ are defined by the equations,

$$(x_{n-1}, \ldots, x_1, x_0)^\rho = (x_{n-2}, \ldots, x_0, \bar{x}_{n-1}), \text{ and}$$

$$(x_{n-1}, \ldots, x_1, x_0)^\delta = (x_{n-1}, \ldots, x_2, \bar{x}_1, \bar{x}_0).$$

A "path" $v_0, v_1, \ldots, v_m$ starting at vertex v_0 and ending at vertex v_m is denoted by a sequence of edge labels $P = p_1 p_2 \cdots p_m$, $p_i \in \{\rho, \rho^{-1}, \delta\}$, such that, $v_i = v_{i-1}^{p_i}$. Using this notation, Leland and Solomon described a routing algorithm that constructed paths of the form:

(a) $g_0 \rho g_1 \rho \cdots g_{n-1}$

(b) $\rho g_0 \rho g_1 \cdots \rho g_{n-1}$

where $g_i \in \{id, \delta\}$.

They proved that any two vertices in $M_n(V, E)$ could be connected by a path of type either (a) or (b) in which $g_i = id$ for at least $\lfloor \frac{n}{2} \rfloor$ values of i. Thus, the diameter of $M_n(V, E)$ was bounded above by $\lfloor \frac{3n}{2} \rfloor$.

Their algorithm had two shortcomings. First, it could not find a path of length less than $n - 1$. As a consequence, a message sent to an adjacent vertex would have to pass through at least $n - 2$ other processors in the network. This would result in unnecessary communication delay and overhead. Also, the routing algorithm failed to use edges labeled ρ^{-1}. This increases the probability of congestion on the other two communication lines.

Our communication scheme solves these problems by computing optimal routes. The first step in developing the algorithm is the observation that the Moebius graph is isomorphic to an QCG.

Definition 7.2 Let G be a finite group generated by H and W, where $H \leq G$ and $W \subseteq G$. The quotient Cayley graph $\Gamma(G, H, W)$ has vertex set $\{Hg | g \in G\}$, and edge set $\{(Hg, Hgw) | g \in G, w \in W\}$.

When defining families of graphs, such as $M_n(V,E)$, one typically specifies the edges by a set of functions (e.g., $\{\rho,\delta\}$). In these cases the following lemma can be a useful characterization of QCGs.

Lemma 7.3 A connected directed graph $\Gamma(V,E)$ is isomorphic to a Quotient Cayley graph if and only if there exists a set of permutations, $S \subseteq Sym(V)$ such that $E = \{(v,v^s)|s \in S,\ v \in V\}$.

Proof: Suppose $\Phi : V \rightarrow \{Hg|g \in G\}$ is an isomorphism between $\Gamma(V,E)$ and $\Gamma(G,H,W)$. For each $w \in W$ define a permutation $s_w \in Sym(V)$, where $v^{s_w} = \Phi^{-1}(\Phi(v)w)$. Let $S = \{s_w|w \in W\}$, then $E = \Phi^{-1}(\{(Hg,Hgw)|w \in W\} = \{(v,v^{s_w})|w \in W\}$.

Suppose $S \subseteq Sym(V)$ such that $E = \{(v,v^s)|s \in S,\ v \in V\}$. Let G denote the subgroup of $Sym(V)$ generated by S, and let $H = G_v$ for some fixed $v \in V$. We know that G is transitive on V since $\Gamma(V,E)$ is connected. Thus, for each $w \in V$ there exists $g \in G$ such that $v^g = w$. There is a natural one-to-one correspondence between V and the right cosets of H in G, namely $\Phi(v^g) = Hg$. This bijection is well defined since $Hg = Hh$ if and only if $v^g = v^h$. Moreover, Φ is an isomorphism between $\Gamma(G,H,S)$ and $\Gamma(V,E)$, since $\Phi(E) = \Phi(\{(v^g,v^{gs})|g \in G,\ s \in S\}) = (\{(Hg,Hgs)|g \in G,\ s \in S\})$. □

Remark 7.4 The isomorphism Φ preserves the edge labels. This fact will be used to reduce the routing problem on $\Gamma(V,E)$ to the problem of finding generating sequences for elements of G.

Since $\rho,\delta \in Sym(V)$ it follows from the proof of Lemma 7.3 that $M_n(V,E)$ is isomorphic to $\Gamma(G^n,H^n,W)$, where $W = \{\rho,\delta\}$, and $H^n = G^n_{\underline{0}}$. We denote the isomorphism

between $M_n(V,E)$ and $\Gamma(G^n, H^n, W)$ by the function $\Phi(v) = H^n g$, where $\overrightarrow{0}^g = v$.

The first thing we must do is decide on a reasonable representation for G^n. If we were to represent a permutation $g \in G^n$ as a product of disjoint cyclics, we would need $O(n2^n)$ space to store g. Instead, we will view G^n as a subgroup of the Affine group A of $(Z_2)^n$, where

$$A = \{(N,v) | N \text{ is an } n \times n \text{ invertible matrix, and } v \in (Z_2)^n\}.$$

For $(N,v) \in A$ and $x \in (Z_2)^n$ the action of (N,v) on x is defined by the equation, $x^{(N,v)} = xN + v$. The product of two elements $(N,v),(L,w) \in A$ is $(N,v)(L,w) = (NL, vL + w)$. This representation allows us to store each $g \in G^n$ in $O(n)$ space.

If we let I be the $n \times n$ identity matrix and let

$$M = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix},$$

then the permutations ρ, ρ^{-1}, and δ can be viewed as elements of A, where

$$\rho = (M, (0, \cdots, 0, 1)),$$

$$\rho^{-1} = (M^{-1}, (1, 0, \cdots, 0)), \text{ and}$$

$$\delta = (I, (0, \cdots, 0, 1, 1)).$$

Definition 7.5 For $0 \leq i \leq n-1$ let $\delta_i = \rho^{-i}\delta\rho^i$. Then $\delta_i = (I, w)$ where $w_j = 0$ unless $j = i+1 \bmod n$ or $j = i \bmod n$. Note that $\delta_0 = \delta$, and

$$\rho^i \delta_{j+i} = \delta_j \rho^i. \tag{VII.12}$$

Convention: Throughout the remainder of the chapter we interpret the subscripts of δ modulo n.

Remark 7.6 Let K^n be the group generated by the set $\{\delta_i | 0 \leq i \leq n-1\}$. Then $K^n \lhd G^n$ and $K^n = \{(I, v) | pr(v) = 0\}$. The group K^n is isomorphic to $(Z_2)^{n-1}$, and any $n-1$ elements from the set $\{\delta_i | 0 \leq i \leq n-1\}$ form a basis for K^n. Since $\Pi_{i=0}^{n-1}\delta_i = (I, \vec{0})$, it follows that every element $k \in K^n$ can be written in exactly two ways as a product of elements from $\{\delta_i | 0 \leq i \leq n-1\}$. We shall say that $v \in (Z_2)^n$ is a characteristic vector of k if $k = \Pi_{v_i=1}\delta_i$. If v is a characteristic vector of k, then the complement of v $(v + \vec{1})$ is the other characteristic vector for k.

Claim 7.7 Let $G \leq A$ defined by,

$$G = \begin{cases} \{(M^i, v) | 0 \leq i \leq n-1,\ v \in (Z_2)^n\} & \text{if } n \geq 3 \text{ and odd} \\ \{(M^i, v) | 0 \leq i \leq n-1,\ v \in (Z_2)^n,\ pr(v) = i \bmod 2\ \} & \text{if } n \geq 2 \text{ and even} \end{cases}$$

then $G^n = G$.

Moreover, if n is odd then each $g \in G^n$ can be written uniquely as a product, rk, where $r \in \langle\rho\rangle$ and $k \in K^n$. If n is even then $g \in G^n$ can be written in exactly two ways as a product, rk, where $r \in \langle\rho\rangle$ and $k \in K^n$.

Proof: Note that $K^n \leq G^n \leq G$. Since $K^n \lhd G^n$, it follows that $\langle\rho\rangle K^n$ is a subgroup of G^n. When n is odd we have $|\langle\rho\rangle K^n| = \frac{|\langle\rho\rangle||K^n|}{|\langle\rho\rangle \cap K^n|} = \frac{2n2^{n-1}}{1} = |G|$. When n is even $|\langle\rho\rangle K^n| = \frac{|\langle\rho\rangle||K^n|}{|\langle\rho\rangle \cap K^n|} = \frac{2n2^{n-1}}{2} = |G|$. □

It is a simple exercise to check that H^n and $\Phi(v)$ are the following:

$$H^n = \begin{cases} \{(M^i, \overrightarrow{0}) | 0 \leq i \leq n-1\} & \text{if } n \geq 3 \text{ and odd} \\ \{(M^i, \overrightarrow{0}) | i = 0, 2, 4, \cdots n-2\} & \text{n even.} \end{cases}$$

$$\Phi(v) = \begin{cases} H^n(I, v) & \text{if } n \geq 3 \text{ and odd} \\ H^n(I, v) & n \text{ even and } pr(v) = 0 \\ H^n(M, v) & n \text{ even and } pr(v) = 1 \end{cases}$$

Viewing the Moebius graph as an QCG has already given us some valuable information about the family of Moebius graphs. In particular, it points out the we are really dealing with two separate families, one for odd values of n, and one for even values of n.

An Optimal Routing Algorithm for the Moebius Graph

In this section we reduce the problem of routing on the Moebius graph to the problem of finding a minimum generating sequence (with respect to $\{\rho, \rho^{-1}, \delta\}$) for $g \in G^n$. Let v_1, v_2 be two vertices from the Moebius graph $M_n(V, E)$. Recall that a sequence of edge labels $P = p_1 p_2 \cdots p_m$, $p_i \in \{\rho, \rho^{-1}, \delta\}$, describes a path from v_1 to v_2 if and only if $v_1^{p_1 p_2 \cdots p_m} = v_2$. Since $\delta^2 = 1$ we may assume that every path in $M_n(V, E)$ has the form

$\rho^{e_{a+1}}\delta\rho^{e_a}\delta\cdots\delta\rho^{e_1}$, where the e_i are integers and nonzero if $2 \leq i \leq a$. The length of the path is $a + \Sigma_{i=1}^{a+1}|e_i|$. Note that a corresponds to the number of δ edges traversed and $\Sigma_{i=1}^{a+1}|e_i|$ to the number of ρ and ρ^{-1} traversed.

Let $OP(v_1, v_2)$ denote an optimal path from v_1 to v_2 in $M_n(V, E)$, and let $MGS(g)$ denote a minimum generating sequence for $g \in G^n$. The algorithm OptimalRoute reduces the problem of computing an optimal route to the problem of finding minimum generating sequence. The input to the algorithm is a pair of vertices from the Moebius graph and the order, n, of the graph. The algorithm returns OP, an optimal path between the vertices.

Procedure OptimalPath(v_1, v_2, n, OP) :

(1) Compute g_1 such that $H^n g_1 = \Phi(v_1)$

(2) Compute g_2 such that $H^n g_2 = \Phi(v_2)$

(3) $OP := MGS(g_1^{-1}g_2)$

(4) For $h \in H^n \setminus \{1\}$ do

(4.1) Path:=$MGS(g_1^{-1}hg_2)$

(4.2) If |Path| < $|OP|$ then Path:=OP

(5) Return(OP)

Claim 7.8 Let $T(g)$ represent the time needed to compute a minimum generating sequence for $g \in G^n$. Given $v_1, v_2 \in M_n(V, E)$ the algorithm OptimalPath finds a shortest path between the vertices in $O(nT(g))$ time.

Proof: The time bound is correct since the running time of algorithm OptimalPath is dominated by the "for loop", and $|H^n| \leq n$.

To prove correctness, recall that the isomorphisim Φ between $M_n(V, E)$ and

$\Gamma(G^n, H^n, W)$ preserves edge labels. Thus, $P = p_1 p_2 \cdots p_r$ is an optimal path from v_1 to v_2 in the Moebius graph if and only if $(H^n g_1)p_1 p_2 \cdots p_r = H^n g_2$, and no word of length less than m maps $H^n g_1$ to $H^n g_2$. We may conclude that any word of minimum length in the set $\{MGS(g_1^{-1} h g_2) | h \in H^n\}$ corresponds to a shortest path from v_1 to v_2. □

We shall now outline the strategy that is used to find a minimum generating sequence for $g \in G^n$. Suppose $MGS(g) = \rho^{e_{a+1}} \delta \rho^{e_a} \delta \cdots \delta \rho^{e_1}$, then using equation (VII.12) we may rewrite g as, $g = \rho^d \delta_{c_a} \cdots \delta_{c_1}$, where $d = \Sigma_{i=1}^{a+1} e_i$, and $c_j = \Sigma_{i=1}^{j} e_i \pmod n$, for $1 \leq j \leq a$. Let v be the characteristic vector of the set $\{c_j | 1 \leq j \leq\}$. Given d and v a minimum generating sequence for g can be constructed using a procedure called QueueWalk.

Of course we are still left with the problem of computing d and v, for $g \in G^n$. We shall show that when n is odd there is only one possible choice for d and therefore, only two possible choices for the vector v (the two vectors will be compliments of each other). When n is even there will be only two possible options for d, and for each value of d there will be two possibilities for v. Our algorithm for computing a minimum generating sequence will call the procedure QueueWalk with all the possible values of d and v, and take the smallest generating sequence found.

The procedure QueueWalk accepts as input a vector $v \in (Z_2)^n$ and an integer d. The procedure returns a sequence $D = d_1, d_2, \ldots, d_m$ and a sequence $T = t_1, t_2, \ldots, t_{wt(v)}$ that satisfy the following conditions:

(a) $d_i \in \{1, -1\}$, and $0 \leq t_1 < t_2 < \ldots < t_{wt(v)} \leq m$

(b) $v_s = 1$ if and only if for some $1 \leq t_j \leq wt(v)$, $\Sigma_{i=1}^{t_j} d_i = s \bmod n$

(c) $\Sigma_{i=1}^{m} d_i = d$

(d) if D' and T' are two sequences satisfying properties 1-3, then $|D| \leq |D'|$

Informally, we may think of v as a circular queue of size n, where cell s of the queue is "marked" if and only if $v_s = 1$. A "walk" on the queue consists of a sequence of steps in either the clockwise (positive) direction, or the counterclockwise (negative) direction. The procedure QueueWalk finds a shortest walk, $D = d_1, d_2, \ldots, d_m$, that starts at cell zero, visits every marked cell of the queue, and terminates a distance $|d|$ away from the start position in the direction $\frac{|d|}{d}$. The length of the walk is m, and the t_i indicate the time at which a marked cell is visited. Note that once a walk is described it is a simple exercise to compute the sequence T in time m.

Procedure QueueWalk(v, d, D, T) :

(1) If $|d| \geq n$, then

(1.1) $d_i := \frac{|d|}{d}$ for $1 \leq i \leq |d|$

(*$m = |d|$*)

(1.2) Compute T satisfying condition (b) above

(1.3) Return(D, T).

(2) If $0 \leq d < n$, then

(2.1) Find a largest block of zeros, (s, t), between $[d + 1, n - 1]$

(*the block of zeros has size $E = t - s - 1$*)

(2.2) $s_1 := n - t$ $s_2 := s - d$

(2.3) $d_i := -1$ for $1 \leq i \leq s_1$

$d_i := 1$ for $s_1 + 1 \leq i \leq 2s_1 + s$

$d_i := -1$ for $2s_1 + s + 1 \leq i \leq 2s_1 + s + s_2$

(*$m = d + 2(n - d - 1 - E)$*)

(2.4) Compute T satisfying condition (b) above

(2.5) Return(D,T).

(3) If $-n < d < 0$, then

(3.1) Find a largest block of zeros, (s,t), between $[1, n - |d| - 1]$

(*the block of zeros has size $E = t - s - 1$*)

(3.2) $s_1 := n - t$ $s_2 := n - |d| - t$

(3.3) $d_i := 1$ for $1 \leq i \leq s$

$d_i := -1$ for $s + 1 \leq i \leq 2s + s_1$

$d_i := 1$ for $2s + s_1 + 1 \leq i \leq 2s + s_1 + s_2$

(*$m = |d| + 2(n - |d| - 1 - E)$*)

(3.4) Compute T satisfying condition (b) above

(3.5) Return(D,T)

Claim 7.9 The procedure QueueWalk terminates in $O(n + |d|)$ time and the sequence $D = d_1, d_2, \ldots, d_m$ constructed by the procedure satisfies conditions (a)-(d).

Proof: We shall assume that the vector v is given as an array indexed from 0 to $n - 1$. The proof consists of the following three cases.

Case I ($|d| \geq n$). The walk specified by the procedure visits every cell, and any walk satisfying $\Sigma_{i=1}^{m} d_i = d$ must of length at least $|d|$. Computing T given D is straightforward.

Case II ($0 \leq d < n$). In line (2.1) of the procedure we find a largest block of consecutive zeros in array v between cell $d + 1$ and cell $n - 1$ (inclusive). We denote this block by (s,t) (i.e., the block starts at cell $s + 1$ and ends at cell $t - 1$). One checks that

the total time needed to execute lines (2.1) through (2.5) is $O(n)$.

To show that the procedure functions correctly we note that the walk specified by the procedure visits all the cells of the queue except cells $s+1$ to $t-1$, which are not marked. Condition (c) is satisfied since $\Sigma_{i=1}^{m} d_i = -s_1 + s_1 + s - s_2 = d$. We shall prove that any walk of shorter length cannot satisfy properties (a)-(c). Just suppose $C = c_1, c_2, \ldots, c_l$ is a walk that satisfies properties (a)-(c) That is, $P_1 \geq \Sigma_{i=1}^{j} c_i$, for $1 \leq j \leq l$. Let P_2 be a smallest partial sum of the C. Then $l \geq 2|P_2| + 2P_1 - d$, and $l < m$ implies that $P_1 - P_2 < n$. Thus, the walk defined by C does not visit cells $P_1 + 1$ to $n + P_2 - 1$, implying that the cells are not marked. However, $l < m$ implies that $t - s < n + P_2 - P_1$, and this implies that (s,t) is not a largest block of zeros.

Case III ($-n < d < 0$) The proof of this case is analogous to the proof of Case II.□

The following claim constructs a generating sequence for $g \in G^n$ using the procedure QueueWalk. The claim proves that with the appropriate input QueueWalk can be used to find a minimum generating sequence for g.

Claim 7.10 Let $g \in G^n$, and suppose $g = \rho^d k$. Let v be a characteristic vector for k and let $D = d_1, d_2, \ldots, d_m$, $T = t_1, t_2, \ldots, t_a$ $(wt(v) = a)$ be the result of a call to QueueWalk(v, d, D, T). Define $p_1 = d_1 + \cdots + d_{t_1}$, $p_{i+1} = d_{t_i+1} + \cdots + d_{t_{i+1}}$ for $1 \leq i < a$, and $p_{a+1} = d_{t_a} + \cdots + d_m$, then $\rho^{p_{a+1}} \delta \rho^{p_a} \delta \cdots \delta \rho^{p_1}$ is a generating sequence for g.

Moreover, if $MGS(g) = \rho^{e_{a+1}} \delta \rho^{e_a} \delta \cdots \delta \rho^{e_1}$, where $d = \Sigma_{i=1}^{a+1} e_i$, and $v_s = 1$ if and only if there exist j such that $\Sigma_{i=1}^{j} e_i = s \bmod n$, then $\rho^{p_{a+1}} \delta \rho^{p_a} \delta \cdots \delta \rho^{p_1}$ is a minimum generating sequence for g.

Proof: First, observe that $\Sigma_{i=1}^{j} p_i \equiv s \bmod n$ if and only if $v_s = 1$. Now using

identity (VII.12) we have $\rho^d k = \rho^{p_{a+1}}\delta\rho^{p_a}\delta\cdots\delta\rho^{p_1}$. To finish the proof it will suffice to show that $\Sigma_{i=1}^{a+1}|p_i| = \Sigma_{i=1}^{a+1}|e_i|$. Since $\rho^{e_{a+1}}\delta\rho^{e_a}\delta\cdots\delta\rho^{e_1}$ is a minimum generating sequence for g it follows that $\Sigma_{i=1}^{a+1}|e_i| \leq \Sigma_{i=1}^{a+1}|p_i|$. Just suppose $\Sigma_{i=1}^{a+1}|p_i| > \Sigma_{i=1}^{a+1}|e_i|$; then we could use the the e_i to define a walk of length $\Sigma_{i=1}^{a+1}|e_i| < m$ for v and d. But by Claim 7.9 such a walk cannot exist. □

Corollary 7.11 (a) If $g = \rho^d k$, and v is the characteristic vector of k, then there exists a generating sequence for g of size

$$\begin{cases} wt(v) + |d| & \text{if } |d| \geq n \\ wt(v) + d + 2(n - d - 1 - E) & \text{if } 0 \leq d < n \\ wt(v) + |d| + 2(n - |d| - 1 - E) & -n < d < 0. \end{cases}$$

Recall that $E = t - s - 1$ (E is defined in $QueueWalk$).

(b) If $MGS(g) = \rho^{e_{a+1}}\delta\rho^{e_a}\delta\cdots\delta\rho^{e_1}$, then $-n \leq \Sigma_{i=1}^{a+1} e_i \leq n$. This follows from the fact that $\rho^{2n} = (I, \overrightarrow{0})$.

The procedure MinGenSeqOdd computes a minimum generating sequence for $g \in G^n$ when n is odd. It accepts as input an element g and an integer n. The procedure QueueWalk is used to construct a generating sequence as described in Claim 7.10.

Procedure MinGenSeqOdd($g = (M^i, v), n, mgs$):

(*We assume that n is odd, and $0 \leq i \leq n - 1$*)

(1) If $i = 0$ and $pr(v) = 0$, then ($g = \rho^0 k$)

(1.1) $d := 0$

(1.2) $k := g$

(2) If $i = 0$ and $pr(v) = 1$, then $(g = \rho^{-n}k)$

(2.1) $d := n$

(2.2) $k := \rho^n g$

(3) If $i \neq 0$ and $pr(v) = i \bmod 2$, then $(g = \rho^i k)$

(3.1) $d := i$

(3.2) $k := \rho^{-i} g$

(4) If $i \neq 0$ and $pr(v) \neq i \bmod 2$, then $(g = \rho^{i-n}k)$

(4.1) $d := i - n$

(4.2) $k := \rho^{n-i} g$

(5) Compute the v and v', the two characteristic vectors for k

(6) QueueWalk(v, d, D, T)

(7) QueueWalk(v', d, D', T')

(8) If $|D| + |T| < |D'| + |T'|$

then use D and T to compute $MGS(g)$

else use D' and T' to compute $MGS(g)$

(9) Return $(mgs := MGS(g))$

Theorem 7.12 Procedure MinGenSeqOdd computes a minimum generating sequence for $g = (M^i, v) \in G^n$ in $O(n)$ time, for odd values of n.

Proof: The time needed to perform any of the first 5 statements is $O(n)$. Statements (6) and (7) take time $O(n + |d|)$ and $|d| \leq n$. If we compute $MGS(g)$ as described in Claim 7.10, then statement (8) takes time $O(n)$.

To finish the proof we must prove that the algorithm computes a minimum gener-

ating sequence for g. Suppose $MGS(g) = \rho^{e_{a+1}}\delta\rho^{e_a}\delta\cdots\delta\rho^{e_1}$, and $c_j = \Sigma_{i=1}^{j} e_i \bmod n$, for $1 \leq j \leq a$. By Corollary 7.11 and Claim 7.7 we have $d = \Sigma_{i=1}^{a+1} e_i$, and it follows that either v or v' is the characteristic vector for $k = \Pi_{j=1}^{a}\delta_{c_j}$. The proof follows from Claim 7.10. □

The procedure for computing a minimum generating sequence for $g \in G^n$ when n is even is analogous to the previous procedure. The only difference is that now there are two possible choices for d (Claim 7.7).

Procedure MinGenSeqEven($g = (M^i, v), n, mgs$):

(*We shall assume that n is even, and $0 \leq i \leq n-1$*)

(1) $k_1 := \rho^{-i}g$ $(g = \rho^i k_1)$

(2) $k_2 := \rho^{n-i}g$ $(g = \rho^{i-n}k_2)$

(3) Compute v_1 and v_1', the characteristic vectors for k_1

(4) Compute v_2 and v_2', the characteristic vectors for k_2

(5) QueueWalk(v_1, i, D_1, T_1)

(6) QueueWalk(v_1', i, D_1', T_1')

(7) QueueWalk($v_2, i-n, D_2, T_2$)

(8) QueueWalk($v_2', i-n, D_2', T_2'$)

(9) If $|D_1'| + |T_1'| < |D_1| + |T_1|$

then set $D_1 := D_1'$ and $T_1 := T_1'$

(10) If $|D_2'| + |T_2'| < |D_2| + |T_2|$

then set $D_2 := D_2'$ and $T_2 := T_2'$

(11) If $|D_1| + |T_1| < |D_2| + |T_2|$

then use D_1 and T_1 to compute $MGS(g)$

else use D_2 and T_2 to compute $MGS(g)$

(12) Return $(mgs := MGS(g))$

Theorem 7.13 Procedure MinGenSeqEven computes a minimum generating sequence for $g = (M^i, v) \in G^n$ in $O(n)$ time for even values of n.

Proof: By the proof of Claim 7.7 we know that for even values of n each $g \in G^n$ can be written in exactly two ways as a product of the form rk, where $r \in \langle \rho \rangle$, and $k \in K^n$. This is done in the first two lines of the procedure. The rest of the proof is analogous to the proof of Theorem 7.12. □

Remark 7.14 The algorithm outlined above for finding a shortest path between two nodes in the Moebius graph is by no means optimal. There are a number of ways that one can improve the running time of the algorithm. For example, since the characteristic vectors (v and v') of k are complements of each other, one can modify the procedure QueueWalk so that it searches simultaneously for a largest block of consecutive "zeros" in v and v'. This will eliminate half of the calls to QueueWalk.

The procedure OptimalPath makes n $(n/2)$ calls to MinGenSeqOdd (MinGenSeqEven). As a result there are n elements from K^n that one must compute. Once one of the elements is obtained, the other $n-1$ elements can be found using Tables 3-6. Of course, these modifications do not change the asymptotic running time, $O(n^2)$, of the algorithm OptimalPath.

Tables 3-6 illustrate the relations between the n elements in K^n that must be found to compute an optimal path from v_1 to v_2 in $M_n(V, E)$. The following notation is used

to define the dependencies. Let $g_1 = \Phi(v_1)$, $g_2 = \Phi(v_2)$, and define k so that $(M^2, \vec{0})k = g_1^{-1}(M^2, \vec{0})g_1$. We denote $\rho^{-i}k\rho^i$, by $k^{(i)}$. Note that if $k = \Pi_{j\in J}\delta_j$, then by equation (VII.12) $k^{(i)} = \Pi_{j\in J}\delta_{j+i}$.

Table 3: For n Odd and $pr(v_1) = pr(v_2)$

For n Odd and $pr(v_1) = pr(v_2)$					
Elements in G^n			Dependencies		
$g_1^{-1}(I, \vec{0})g_2$	$=$	$\rho^0 k_0$			
$g_1^{-1}(M^2, \vec{0})g_2$	$=$	$\rho^2 k_2$	$\delta_0 k^{(0)} k_0$	$=$	k_2
$g_1^{-1}(M^4, \vec{0})g_2$	$=$	$\rho^4 k_4$	$\delta_2 k^{(2)} k_2$	$=$	k_4
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-1}, \vec{0})g_2$	$=$	$\rho^{n-1} k_{n-1}$	$\delta_{n-3} k^{(n-3)} k_{n-3}$	$=$	k_{n-1}
$g_1^{-1}(M^1, \vec{0})g_2$	$=$	$\rho^{1-n} k_1$	$\delta_{n-1} k^{(n-1)} k_{n-1}$	$=$	k_1
$g_1^{-1}(M^3, \vec{0})g_2$	$=$	$\rho^{3-n} k_3$	$\delta_1 k^{(1)} k_1$	$=$	k_3
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-4}, \vec{0})g_2$	$=$	$\rho^{-4} k_{n-4}$	$\delta_{n-6} k^{(n-6)} k_{n-6}$	$=$	k_{n-4}
$g_1^{-1}(M^{n-2}, \vec{0})g_2$	$=$	$\rho^{-2} k_{n-2}$	$\delta_{n-4} k^{(n-4)} k_{n-4}$	$=$	k_{n-2}
			$\delta_{n-2} k^{(n-2)} k_{n-2}$	$=$	k_0

The Diameter of the Moebius Graph

Leland and Solomon established an upper bound of $\lfloor \frac{3n}{2} \rfloor$ for the Moebius graph of order n. They also proved (by computer search) that the actual diameter of the $M_n(V, E)$ was $\lceil \frac{3n}{2} \rceil - 2$, for $2 \leq n \leq 11$. In this section we prove that the diameter of the Moebius graph $M_n(V, E)$ is $\lceil \frac{3n}{2} \rceil - 2$ when n is odd, and $\frac{3n}{2} - 1$ when n is even and greater than 10. Throughout the remainder of this chapter we shall use the notation from Tables 3-6.

Lemma 7.15 If n is odd and v_1 and v_2 are two vertices from the $M_n(V, E)$, then there exists a path between the vertices of length no more than $\lceil \frac{3n}{2} \rceil - 2$.

Table 4: For n Odd and $pr(v_1) \neq pr(v_2)$

For n Odd and $pr(v_1) \neq pr(v_2)$					
Elements in G^n			Dependencies		
$g_1^{-1}(I, \vec{1})g_2$	$=$	$\rho^1 k_1$			
$g_1^{-1}(M^3, \vec{0})g_2$	$=$	$\rho^3 k_3$	$\delta_1 k^{(1)} k_1$	$=$	k_3
$g_1^{-1}(M^5, \vec{0})g_2$	$=$	$\rho^5 k_5$	$\delta_3 k^{(3)} k_3$	$=$	k_5
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-2}, \vec{0})g_2$	$=$	$\rho^{n-2} k_{n-2}$	$\delta_{n-4} k^{(n-4)} k_{n-4}$	$=$	k_{n-2}
$g_1^{-1}(I, \vec{0})g_2$	$=$	$\rho^n k_0$	$\delta_{n-2} k^{(n-2)} k_{n-2}$	$=$	k_0
$g_1^{-1}(M^2, \vec{0})g_2$	$=$	$\rho^{2-n} k_2$	$\delta_0 k^{(0)} k_0$	$=$	k_2
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-3}, \vec{0})g_2$	$=$	$\rho^{-3} k_{n-3}$	$\delta_{n-5} k^{(n-5)} k_{n-5}$	$=$	k_{n-3}
$g_1^{-1}(M^{n-1}, \vec{0})g_2$	$=$	$\rho^{-1} k_{n-1}$	$\delta_{n-3} k^{(n-3)} k_{n-3}$	$=$	k_{n-1}
			$\delta_{n-1} k^{(n-1)} k_{n-1}$	$=$	k_1

Table 5: For n Even and $pr(v_1) \neq pr(v_2)$

For n Even and $pr(v_1) \neq pr(v_2)$					
Elements in G^n			Dependencies		
$g_1^{-1}(M^0, \vec{0})g_2$	$=$	$\rho^1 k_1$			
$g_1^{-1}(M^2, \vec{0})g_2$	$=$	$\rho^3 k_3$	$\delta_1 k^{(1)} k_1$	$=$	k_3
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-2}, \vec{0})g_2$	$=$	$\rho^{n-1} k_{n-1}$	$\delta_{n-3} k^{(n-3)} k_{n-3}$	$=$	k_{n-1}
$g_1^{-1}(M^0, \vec{0})g_2$	$=$	$\rho^{1-n} k_{1-n}$	$\delta_{n-1} k^{(n-1)} k_{n-1}$	$=$	k_{1-n}
$g_1^{-1}(M^2, \vec{0})g_2$	$=$	$\rho^{3-n} k_{3-n}$	$\delta_1 k^{(1)} k_{1-n}$	$=$	k_{3-n}
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-4}, \vec{0})g_2$	$=$	$\rho^{-3} k_{-3}$	$\delta_{n-3} k^{(n-3)} k_{-5}$	$=$	k_{-3}
$g_1^{-1}(M^{n-2}, \vec{0})g_2$	$=$	$\rho^{-1} k_{-1}$	$\delta_{n-3} k^{(n-3)} k_{-3}$	$=$	k_{-1}
			$\delta_{n-1} k^{(n-1)} k_{-1}$	$=$	k_1

Table 6: For n Even and $pr(v_1) = pr(v_2)$

For n Even and $pr(v_1) = pr(v_2)$					
Elements in G^n			Dependencies		
$g_1^{-1}(M^0, \vec{0})g_2$	$=$	$\rho^0 k_0$			
$g_1^{-1}(M^2, \vec{0})g_2$	$=$	$\rho^2 k_2$	$\delta_0 k^{(0)} k_0$	$=$	k_2
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-2}, \vec{0})g_2$	$=$	$\rho^{n-2} k_{n-2}$	$\delta_{n-4} k^{(n-4)} k_{n-4}$	$=$	k_{n-2}
$g_1^{-1}(M^0, \vec{0})g_2$	$=$	$\rho^{-n} k_{-n}$	$\delta_{n-2} k^{(n-2)} k_{n-2}$	$=$	k_{-n}
$g_1^{-1}(M^2, \vec{0})g_2$	$=$	$\rho^{2-n} k_{2-n}$	$\delta_0 k^{(0)} k_{-n}$	$=$	k_{2-n}
	$\vdots$			$\vdots$	
$g_1^{-1}(M^{n-4}, \vec{0})g_2$	$=$	$\rho^{-4} k_{-4}$	$\delta_{n-6} k^{(n-6)} k_{-6}$	$=$	k_{-4}
$g_1^{-1}(M^{n-2}, \vec{0})g_2$	$=$	$\rho^{-2} k_{-2}$	$\delta_{n-4} k^{(n-4)} k_{-4}$	$=$	k_{-2}
			$\delta_{n-2} k^{(n-2)} k_{-2}$	$=$	k_0

Proof: Let $\Phi(v_1) = H^n g_1$ and $\Phi(v_2) = H^n g_2$. By the definition of Φ we have $g_1 = (I, v_1)$ and $g_2 = (I, v_2)$. It will suffice (by claim 7.8) to prove that there exists $h \in H^n$ such that $g_1^{-1} h g_2$ has a generating sequence of length no more than $\lceil \frac{3n}{2} \rceil - 2$.

Case I ($pr(v_1) = pr(v_2)$) Let $g_1^{-1}(M^{n-1}, \vec{0})g_2 = \rho^{n-1} k_{n-1}$ (notation from Table 3). By Remark 7.6 we know that there exists a characteristic vector, v, for k_{n-1} such that $wt(v) \leq \lceil \frac{n}{2} \rceil - 1$. Now, by Corollary 7.11 we have,

$$|MGS(\rho^{n-1} k_{n-1})| \leq wt(v) + n - 1 + 2(0) \leq \lceil \frac{3n}{2} \rceil - 2.$$

CaseII($pr(v_1) \neq pr(v_2)$) Let $g_1^{-1}(M^{n-2}, \vec{0})g_2 = \rho^{n-2} k_{n-2}$. By Remark 7.6 we know that there exists a characteristic vector, v for k_{n-2}, such that $v_{n-1} = 0$. If the $wt(v) \leq \lceil \frac{n}{2} \rceil$, then

$$|MGS(\rho^{n-2} k_{n-2})| \leq wt(v) + n - 2 + 2(0) \leq \lceil \frac{3n}{2} \rceil - 2.$$

On the other hand if $wt(v) > \lceil \frac{n}{2} \rceil$, then let v' be the complement of v. We know that v' is a characteristic vector of k_{n-2} and $wt(v') \leq \lceil \frac{n}{2} \rceil - 2$. Thus,

$$|MGS(\rho^{n-2}k_{n-2})| \leq wt(v') + n - 2 + 2(1) \leq \lceil \frac{3n}{2} \rceil - 2.$$

□

Theorem 7.16 If n is odd then the diameter of $M_n(V,E)$ is $\lceil \frac{3n}{2} \rceil - 2$.

Proof: Let $v_1 = (0,1,0,1,0,\ldots,0,1,0)$ and let $v_2 = (1,0,1,0,1,\ldots,1,0,1,0,0)$. If $\Phi(v_1) = H^n g_1$ and $\Phi(v_2) = H^n g_2$, then by the definition of Φ, $g_1 = (I, v_1)$ and $g_2 = (I, v_2)$. By Lemma 7.15 and Claim 7.8 it suffices to show that $|MGS(g_1^{-1} h g_2)| \geq \lceil \frac{3n}{2} \rceil - 2$, for all $h \in H^n$.

Since $pr(v_1) = pr(v_2)$ we have for $0 \leq i \leq n-1$, $(I, v_1)(M^i, \vec{0})(I, v_2) = \rho^d k_d$, where $d = i$ if i is even, and $d = i - n$ if i is odd (d must be even). Solving for k_d we find $k_d = (I, (1,1,\ldots,1,1,0))$. Let v and v' be the characteristic vectors of k_d, where $v_j = 0$ if and only if $j \equiv 0 \bmod 2$, and $v'_j = 0$ if and only if $j \equiv 1 \bmod 2$.

By Corollary 7.11 we have,

$$|MGS(\rho^d k_d)| \geq \begin{cases} \lceil \frac{n}{2} \rceil - 1 + d + 2(n - d - 1 - E) & \text{if } 0 \leq d < n \\ \lceil \frac{n}{2} \rceil - 1 + |d| + 2(n - |d| - 1 - E) & -n < d < 0. \end{cases}$$

Next we observe that if $|d| = n - 1$, then $E = 0$, otherwise $E = 1$. □

Fact 7.17 Let $v \in (Z_2)^n$, then $(I, v + vM^i) \in K^n$. Moreover, if i, n are both even and $(I, v + vM^i) = \Pi_{j \in J} \delta_j$, then $|J|$ is even.

Proof: The proof is by induction on $wt(v)$. Let $v = (x_{n-1}, \ldots, x_0)$. If the $wt(v) = 0$, then $J = \emptyset$ or $J = \{0, 1, \ldots n-1\}$. If $wt(v) = 1$ and $x_s = 1$, then $J = \{s, s+1, \ldots, s+i-1\}$ or J is equal to the complement of this set (relative to $\{0, 1, \ldots n-1\}$). Suppose that $wt(v) \geq 2$. Let $v = u + w$, where $wt(u) < wt(v)$, and let $wt(w) < wt(v)$. Define J_1 and J_2, so that $(I, u + uM^i) = \Pi_{j \in J_1} \delta_j$, and $(I, w + wM^i) = \Pi_{j \in J_2} \delta_j$. Then $(I, v + vM^i) = \Pi_{j \in J_1 \triangle J_2} \delta_j$, and we have $J = J_1 \triangle J_2$ ($\triangle$ stands for symmetric difference). By induction we may assume that $|J_1|$ and $|J_2|$ are even, and thus, $|J|$ is even. □

Lemma 7.18 Let n be even, and let v_1 and v_2 be two vertices in the Moebius graph $M_n(V, E)$. If $pr(v_1) \neq pr(v_2)$, then there exists a path between the vertices of length no more than $\frac{3n}{2} - 2$.

Proof: We assume, without loss of generality, that $pr(v_1) = 0$ and $pr(v_2) = 1$. Let $g_1 = (I, v_1)$ and $g_2 = (M, v_2)$. It will suffice to show that there exists $h \in H^n$, such that $g_1^{-1} h g_2$ has a generating sequence of length no more than $\frac{3n}{2} - 2$. Let $g_1^{-1}(M^{n-2}, \vec{0}) g_2 = \rho^{n-1} k_{n-1}$, and let v be a characteristic vector of k_{n-1} such that $wt(v) \leq \frac{n}{2}$. If $wt(v) \neq \frac{n}{2}$ then we have, via Corollary 7.11,

$$|MGS(\rho^{n-1} k_{n-1})| \leq \frac{n}{2} - 1 + n - 1.$$

On the other hand if $wt(v) = \frac{n}{2}$, then consider $g_1^{-1}(I, \vec{0}) g_2 = \rho^{1-n} k_{1-n}$. By Table 5 we have,

$$k_{1-n} = \delta_{n-1} k^{(n-1)} k_{n-1}.$$

Let w be the characteristic vector for k_{1-n} and recall that $k = (I, v_1 + v_1 M^2)$. Using Fact 7.17 we conclude that the $wt(w) \neq \frac{n}{2}$. Without loss of generality we may assume that $wt(v) < \frac{n}{2}$ and

$$|MGS(\rho^{1-n} k_{1-n})| \leq \frac{n}{2} - 1 + n - 1.$$

□

Lemma 7.19 Let n be even, and let v_1 and v_2 be two vertices in the Moebius graph $M_n(V, E)$. If $pr(v_1) = pr(v_2)$, then there exists a path between the vertices of length no more than $\frac{3n}{2} - 1$.

Proof: Let $\Pi(v_1) = H^n g_1$ and $\Pi(v_2) = H^n g_2$. If $pr(v_1) = pr(v_2) = 0$, then $g_1 = (I, v_1)$ and $g_2 = (I, v_2)$. If $pr(v_1) = pr(v_2) = 1$, then $g_1 = (M, v_1)$ and $g_2 = (M, v_2)$. It will suffice to show that there exists $h \in H^n$, such that $g_1^{-1} h g_2$ has a generating sequence of length no more than $\frac{3n}{2} - 1$. Let $g_1^{-1}(I, \overrightarrow{0}) g_2 = \rho^{-n} k_{-n}$, and let v be a characteristic vector of k_{-n} such that $wt(v) \leq \frac{n}{2}$. If $wt(v) \neq \frac{n}{2}$ then we have,

$$|MGS(\rho^{-n} k_{-n})| \leq \frac{n}{2} - 1 + n.$$

If the $wt(v) = \frac{n}{2}$, then we consider the element $g_1^{-1}(M^2, \overrightarrow{0}) g_2 = \rho^{2-n} k_{2-n}$, and let w be a characteristic vector of k_{2-n}. As before, we use Table 6 and Fact 7.17 to prove that $wt(w) \neq \frac{n}{2}$ ($k_{2-n} = \delta_0 k^{(0)} k_{-n}$). So we may assume that $wt(w) \leq \frac{n}{2} - 1$; and it follows that,

$$|MGS(\rho^{2-n} k_{2-n})| \leq \frac{n}{2} - 1 + |2 - n| + 2(1).$$

□

Theorem 7.20 Let n be an even integer greater than 10. Then the diameter of $M_n(V,E)$ is $\frac{3n}{2}-1$.

Proof: By Lemma 7.18 and Lemma 7.19 it will suffice to exhibit $v_1, v_2 \in V$ such that $dist(v_1,v_2) = \frac{3n}{2}$. Let $g_1 = \Phi(v_1)$, $g_2 = \Phi(v_2)$; we shall show that each word in the set $\{MGS(g_1^{-1}hg_2)|h \in H^n\}$ has length at least $\frac{3n}{2}-1$. Recall that we are using notation from Table 6. In particular, $k = (I, v_1M^2 + v_1)$, and the elements in K^n that we are interested in are $k_{-n}, k_{2-n}, \ldots, k_{n-2}$. The proof will consist of 4 cases.

Case I ($n = 12$) Let $v_1 = 0^5 1^7$ and let $v_2 = 001000101101$. One checks, by brute force, that $dist(v_1,v_2) = \frac{3n}{2}-1$. For the reader's convenience we have listed in Table 7 the characteristic vectors for $k_{-12}, k_{-10}, \ldots, k_{10}$. The following facts were used to construct the Table: $k = \delta_0\delta_7$, $k_{-12} = \delta_{10}\delta_7\delta_4\delta_3\delta_1\delta_0$, and $k_{j+2} = k_j\delta_{7+j}$. Now using Corollary 7.11

Table 7: Characteristic Vectors for $n = 12$

k_j	Characteristic vector for k_j
k_{-12}	010010011011
k_{-10}	010000011011
k_{-8}	011000011011
k_{-6}	111000011011
k_{-4}	111000011001
k_{-2}	111000010001
k_0	111000110001
k_2	111010110001
k_4	110010110001
k_6	010010110001
k_8	010010110011
k_{10}	010010111011

one checks that $dist(v_1,v_2) = 17$.

Case II ($n = 14$) Let $v_1 = 0^6 1^8$, and let $v_2 = 00010011011100$. Now one may check, by brute force, that $dist(v_1, v_2) = \frac{3n}{2} - 1$. Note that $k = \delta_0\delta_8$, $k_{-14} = \delta_{12}\delta_{10}\delta_9\delta_7\delta_5\delta_4\delta_2$ and that $k_{j+2} = k_j\delta_{8+j}$. We use this information to build Table 8. Once again Corollary

Table 8: Characteristic Vectors for $n = 14$

k_j	Characteristic vector for k_j
k_{-14}	01011010110100
k_{-12}	01011110110100
k_{-10}	01001110110100
k_{-8}	00001110110100
k_{-6}	00001110110101
k_{-4}	00001110110001
k_{-2}	00001110100001
k_0	00001111100001
k_2	00001011100001
k_4	00011011100001
k_6	01011011100001
k_8	01011011100000
k_{10}	01011011100100
k_{12}	01011011110100

7.11 can be used to verify that $dist(v_1, v_2) = 20$.

For the last two cases we shall find $v_1, v_2 \in V$ so that any characteristic vector v_j for k_j satisfies the following properties:

(a) $wt(v_j) \geq \frac{n}{2} - 1$

(b) no string of digits in v_j has length more than 5

By Corollary 7.11 we conclude that $|MGS(\rho^j k_j)| \geq \frac{3n}{2} - 1$ whenever $|j| \leq n - 12$. Thus, we need only check (by hand) that $|MGS(\rho^j k_j)| \geq \frac{3n}{2} - 1$ for $j = n - 10, n - 8, \ldots, n - 2, n, 2 - n, \ldots, 10 - n$.

Case III ($n = 16 + 4m$) Let $v_1 = 0^{n-8}1^8$, and let $v_2 = 00101100(1010)^m11011100$. Note that $k = \delta_0\delta_8$, $k_{j+2} = k_j\delta_{8+j}$, and the characteristic vector for k_{-n} is $k_{-n} = 10110001(0011)^m01001011$. It is a simple (but tedious) task to check that properties (a) and (b) hold. Now Corollary 7.11 and Table 9 may be used to check that $|MGS(\rho^j k_j)| \geq \frac{3n}{2}$ for the remaining 11 values in question.

Table 9: Characteristic Vectors for $n = 16 + 4m$

k_j	Characteristic vector for k_j
k_{-n}	$10110001(0011)^m01001011$
k_{2-n}	$1011000?(001?)^m01001011$
k_{4-n}	$10110?0?(0?1?)^m01001011$
k_{6-n}	$101?0?0?(0?1?)^m01001011$
k_{8-n}	$1?1?0?0?(0?1?)^m01001011$
k_{10-n}	$1?1?0?0?(0?1?)^m0100101?$
$\vdots$	$\vdots$
k_{n-10}	$11110001(0011)^m00011110$
k_{n-8}	$10110001(0011)^m00011110$
k_{n-6}	$10110001(0011)^m00011111$
k_{n-4}	$10110001(0011)^m00011011$
k_{n-2}	$10110001(0011)^m00001011$

Case IV ($n = 18 + 4m$) Let $v_1 = 0^8 1^{n-8}$, and let $v_2 = 001011001100(1010)^m101100$. Note that $k = \delta_{n-8}\delta_8$, $k_{j+2} = k_j\delta_{j-8}$, and the characteristic vector for k_{-n} is $010011101011(1001)^m100100$. Now Corollary 7.11 and Table 10 may be used to to finish the proof. □

Table 10: Characteristic Vectors for $n = 18 + 4m$

k_j	Characteristic vector for k_j
k_{-n}	$010011101011(1001)^m100100$
k_{2-n}	$010011111011(1001)^m100100$
k_{4-n}	$010010111011(1001)^m100100$
k_{6-n}	$010110111011(1001)^m100100$
k_{8-n}	$000110111011(1001)^m100100$
k_{10-n}	$000110111011(1001)^m100101$
$\vdots$	$\vdots$
k_{n-10}	$010011101110(1?0?)^m1?0?0?$
k_{n-8}	$010011101110(1?0?)^m1?0?00$
k_{n-6}	$010011101110(1?01)^m1?0100$
k_{n-4}	$010011101110(1001)^m100100$
k_{n-2}	$010011101111(1001)^m100100$

CHAPTER VIII

SUMMARY AND FUTURE WORK

In this dissertation we have focused our attention on bases, SGSs and subgroup towers for permutation groups. We investigated both the sequential and parallel complexity of several algebraic problems involving bases and SGSs. We have also shown how subgroup towers and SGSs can be used to design dense interconnection networks that are accompanied by efficient routing algorithms.

In Chapter II we answered in the negative the question asked by Finkelstein as to whether or not the Greedy1 algorithm always computes a minimum base. In fact, we proved that the problem of computing a minimum base for $G \leq Sym(n)$ is NP-hard. Moreover, the problem remains NP-hard even if we restrict G to be an abelian group with orbits of size no more than 8.

For abelian groups with orbits of size 7 or less we described a polynomial time algorithm for computing minimum bases. Thus, for abelian groups this bound on the size of the orbits is sharp. The computational complexity of computing minimum bases for arbitrary groups with orbits of size less than 8 remains open. We have preliminary results that reduce this problem to the cases where the orbits have size 4, 6 and 7.

In Chapter III we examined the problem of approximating minimum bases for permutation groups. We observed that it was possible for $G \leq Sym(n)$ to have a nonre-

dundant base of size $\frac{1}{3}\mathcal{M}(G)\log n$. In contrast, the Greedy1 algorithm always produces a base of size no more than $\lceil \mathcal{M}(G)\log\log n\rceil + \mathcal{M}(G)$. We went on to prove that, up to a constant, this bound on the size of a Greedy1 base is sharp. That is, for any n sufficiently large there exists $G \leq Sym(n)$, such that every Greedy1 base for G has size at least $\frac{1}{5}\mathcal{M}(G)\log\log n$.

We examined a second greedy algorithm, Greedy2, for constructing small bases. Although this algorithm works well in a number of cases we proved that its worst case performance approaches the upper bound $O(\mathcal{M}(G)\log n)$. In particular, we showed that for fixed ϵ, $0 < \epsilon < 1$, and for any n sufficiently large, there exists $G \leq Sym(n)$ such that every Greedy2 base for G has size at least $\frac{1}{6}\mathcal{M}(G)(\log n)^{1-\epsilon}$.

The approximation algorithm with the best worst case performance that we know of is the Greedy1 algorithm. Of course, there are other natural greedy heristics that one could consider. For example, we could pick a point that minimizes the number of nontrivial G-orbits. We do not anticipate some other greedy algorithm having a better worst case performance than Greedy1. To improve on the Greedy1 algorithm we believe it will be necessary to take into account the structure of the groups involved.

The question is still open as to whether or not a "good" parallel algorithm exists for approximating minimum bases. In Chapter IV we proved that DGB is P-complete. This implies that it is unlikely that a parallel version of the Greedy1 algorithm exists. In this chapter we also examined FAC, the problem of factoring in parallel an element $g \in G$ through a given SGS for G. The question, "Is FAC $\in$ NC?", was stated as an open problem in [4]. Under the assumption that P $\neq$ NC we answer this question in the negative by proving that FAC is P-complete. The final result in Chapter IV showed that

one could find, in parallel, an NC-efficient SGS for any polynomial tower of a solvable group. Can this result be extended to a larger class of groups?

In Chapters V, VI and VII we applied the algebraic techniques developed for permutation groups to the study of interconnection networks. Our emphasis was not only on the construction of Cayley networks but also on the routing algorithms that are needed to utilize the networks. It was in the development of efficient failsoft routing algorithms that subgroup towers and SGSs palyed a key role.

In Chapter V we proved that any undirected SGS Cayley network has a failsoft routing algorithm for computing two disjoint paths between any two nodes in the network. We showed that for a directed SGS Cayley network $\Gamma(G,W)$, constructed from a normal subgroup tower, it was possible to find $|W|$ disjoint paths between any two nodes.

In this chapter we also showed how Valiant's permutation routing algorithm could be adapted to run on directed SGS Cayley networks. In fact, the algorithm solves the partial permutation routing problem. This is one of three subroutines needed in the simulation of idealistic (PRAM) computers by realistic (multiprocessor network) computers [43, page 227]. The other two subroutines are sorting and distribution. One of the problems we are working on now is an efficient sorting algorithm for SGS Cayley networks. We hope to show that the SGS Cayley networks can use a modified version of odd-even merge sort.

There are two other questions concerning permutation routing that warrant further investigation. First, Pippenger has described a network in which a variant of Valiant's algorithm performs permutation routing and uses bounded queues [38]. Can this algorithm be adapted to SGS Cayley networks? Second, Leighton, Maggs and Rao have described an off-line algorithm that eliminates the probabilistic component of permutation routing

[28]. Is there an on-line version of this algorithm that will run on SGS Cayley networks?

In Chapter VI we extended Faber's work on universal broadcast schemes. We proved that it is possible to find an optimal universal broadcast algorithm for a number of Cayley networks. In particular, we showed that if there is an optimal universal broadcast for $\Gamma(G,W)$ and $\lfloor\frac{|G|-1}{|W|}\rfloor - 1 > |W| > 2$, then there is an optimal universal broadcast for $\Gamma(H,Y)$, where $H = G \times \langle w_{d+1}\rangle$ and $Y = \{(w,id)|w \in W\} \cup \{(id, w_{d+1})\}$ ($|\langle w_{d+1}\rangle| \geq 2$).

As a consequence of this result we proved that if G is an abelian group and $W = \{w_1, w_2, \ldots, w_k\}$ is a generating set for G such that $w_i \notin \langle w_1, w_2, \ldots w_{i-1}\rangle$, $1 < i \leq k$, then the time needed for a universal broadcast in $\Gamma(G,W)$ is $\lceil\frac{|G|-1}{|W|}\rceil$. This yields an alternate proof of Vaber's result that the time for a universal broadcast in a d-cube is $\lceil\frac{(2^d-1)}{d}\rceil$.

We also proved that an optimal universal broadcast can be found for the Cayley network $\Gamma(G,W)$, where $G = {}^k Z_2$ and W is the canonical set of minimal generators defined in Example 5.9. Recently we described a universal broadcast that runs in time $\lceil\frac{|G|-1}{|W|}\rceil$ on the Cayley network $\Gamma(G,W')$, where $G = {}^k Z_2$ and W' is the canonical set of generators defined in Example 5.7. We would like to extend these results to other nonabelian Cayley networks.

In Chapter VII we used QCGs to analyze nonsymmetric networks. Our first result was a useful characterization of QCGs. We proved that a connected directed graph is isomorphic to a QCG if and only if it satisfies a labeling property. This result was used to prove that the Moebius graph is isomorphic to a QCG. The QCG was used to describe an efficient minimum routing algorithm for the Moebius graph and to determine the true diameter of the graph.

The groups that defined the QCG played a key role in solving these two open prob-

lems. Not only did the groups guide the development of our minimum routing algorithm, but they also played an important role in proving that the algorithm was correct.

BIBLIOGRAPHY

[1] ARDEN, B., AND LEE, H. Analysis of Chordal Ring networks. *IEEE Trans. Electron. Comput. C-30* (Apr. 1981), 291–301.

[2] BABAI, L. On the order of uniprimitive permutation groups. *Annals of Math. 113* (1981), 553–568.

[3] BABAI, L. On the length of subgroup chains in the symmetric group. *Communications in Algebra 14* (1986), 1729–1736.

[4] BABAI, L., LUKS, E., AND SERESS, A. Permutation groups in NC. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (1987), vol. 19, pp. 409–420.

[5] BANNAI, E., AND ITO, T. On finite Moore graphs. *Journal of Fac. Sci. Univ. Tokyo 20* (1973), 191–208.

[6] BROWN, C., FINKELSTEIN, L., AND PURDOM, P. Efficient implementation of Jerrum's algorithm for permutation groups. Pre-print.

[7] BROWN, C., FINKELSTEIN, L., AND PURDOM, P. Backtrack searching in the presence of symmetry. Tech. Rep. NU-CCS-87-2, Northeastern University, 1987.

[8] CAMERON, P. Personal correspondence to K. D. Blaha.

[9] CANNON, J. A computational toolkit for finite permutation groups. In *Proceedings of Rutgers Group Theory, 1983-1984* (1984), pp. 1–18.

[10] CARLSSON, G., CRUTHIRDS, J., SEXTON, H., AND WRIGHT, C. Interconnection networks based on a generalization of Cube-connected cycles. *IEEE Trans. on Comput. C-34 No. 8* (Aug. 1985), 769–722.

[11] CARLSSON, G., FELLOWS, M., SEXTON, H., AND WRIGHT, C. Group theory as an organizing principle in parallel processing. Pre-print.

[12] CARLSSON, G., SEXTON, H., AND WRIGHT, C. Cayley networks and generalized Cube-connected cycles. Pre-print.

[13] CHERNOFF, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Math. Stat. 23* (1952), 493–507.

[14] COOK, S. A. A taxonomy of problems with fast parallel algorithms. *Information and Control 64* (1985), 2–22.

[15] DAWES, R., AND MEIJER, H. Arc-minimal digraphs of specified diameter. Pre-print.

[16] DOTY, K. New design for dense processor interconnection networks. *Trans. Electron. Comput. C-33 No. 5* (1984), 447–450.

[17] EVEN, S., AND GOLDREICH, O. The minimal-length generating sequence problem is NP-hard. *Journal of Algorithms 2* (1980), 311–313.

[18] FABER, V. Global communication algorithms for hypercubes and other Cayley coset graphs. Pre-print.

[19] FINKELSTEIN, L. Personal correspondence to E. M. Luks.

[20] FURST, M., HOPCROFT, J., AND LUKS, E. M. Polynomial-time algorithms for permutation groups. In *Proceedings 21st Annual Symposium on Foundations of Computer Science* (1980), vol. 21, pp. 36–41.

[21] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.

[22] HOEFFDING, W. On the distribution of the number of successes in independent trials. *Annals of Math. Stat. 27* (1956), 713–721.

[23] HOPCROFT, J., AND ULLMAN, J. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Pub. Co., Menlo Park, Calif., 1979.

[24] JERRUM, M. The complexity of finding minimal-length generating sequence. Tech. Rep. CRS-139-83, Univ. of Edinburgh, 1983.

[25] JERRUM, M. A compact representation for permutation group. *Journal of Algorithms 7* (1986), 60–78.

[26] KNUTH, D. Efficient representation of perm groups. Pre-print.

[27] LAWLER, E. L. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[28] LEIGHTON, T., MAGGS, B., AND RAO, S. Universal packet routing algorithms. In *Proceedings 29th Annual Symposium on Foundations of Computer Science* (1988), vol. 29, pp. 256–269.

[29] LELAND, W. Density and reliability of interconnection topologies for multicomputers. Tech. Rep. CS-TR-478, Univ. of Wisconsin-Madison, 1982.

[30] LELAND, W., AND SOLOMON, M. Dense trivalent graphs for processor interconnection. *Trans. Electron. Comput. C-31 No 3* (1982), 219–222.

[31] LOVÁSZ, L. The matroid matching problem. In *Proceedings of the Conference on Algebraic Methods in Graph Theory* (1978).

[32] LOVÁSZ, L. Matroid matching and some applications. *Journal of Combin. Theory Ser. B 28* (1980), 208–236.

[33] LOVÁSZ, L., AND PLUMMER, M. *Matching Theory.* North-Holland, Amsterdam, 1986.

[34] LUKS, E. Parallel algorithms for permutation groups and graph isomorphism. In *Proceedings 27th Annual Symposium on Foundations of Computer Science* (1986), vol. 27, pp. 292–302.

[35] LUKS, E., AND MCKENZIE, P. Fast parallel computation with permutation groups. In *Proceedings 26th Annual Symposium on Foundations of Computer Science* (1985), vol. 26, pp. 505–514.

[36] HALL, M., JR. *The Theory of Groups.* Macmillan, New York, 1959.

[37] MCKENZIE, P., AND COOK, S. The parallel complexity of abelian permutation group problems. Tech. Rep. No. 181-85, Dept. of Computer Science, University of Toronto, 1985.

[38] PIPPENGER, N. Parallel communication with limited buffers. Tech. Rep., IBM Research Laboratory, San Jose, Calif., 1984.

[39] PREPARATA, F., AND VUILLEMIN, J. The Cube-connected cycles: A versatile network for parallel computation. *Commun. Ass. Comput. Mach. 24* (1980), 300–309.

[40] SEITZ, C. The CosmicCube. *Commun. of the ACM 28* (1985), 22–33.

[41] SIMS, C. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra* (1970), J. Leech, Ed., Pergamon Press, pp. 169–183.

[42] SIMS, C. Determining the conjugacy classes of a permutation group. In *Computers in Algebra and Number Theory* (1970), G. Birkhoff and J. M. Hall, Eds., vol. 4, pp. 191–195.

[43] ULLMAN, J. *Computational Aspects of VLSI.* Computer Science Press, Rockville, Maryland, 1984.

[44] VALIANT, L. A scheme for fast parallel communication. *SIAM Journal of Comput. 11* (1982), 350–361.

[45] VALIANT, L., AND BREBNER, G. Universal schemes for parallel communication. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing* (1981), vol. 13, pp. 263–277.

[46] VON CONTA, C. Torus and other networks as communication networks with up to some hundred points. *Trans. Electron. Comput.* 7 (1983), 657–666.

[47] WIELANDT, H. *Finite Permutation Groups.* Academic Press, New York-London, 1964.