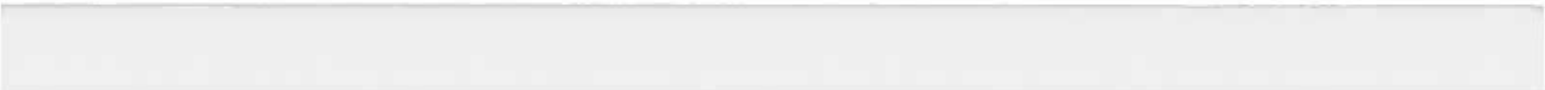

**A Greedy Approach
to a NP-hard Problem
for Permutation Groups**

Kenneth D. Blaha

CIS-TR-86-16
May 4, 1988

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE
UNIVERSITY OF OREGON



A Greedy Approach to a NP-hard Problem for Permutation Groups

Kenneth D. Blaha*

Department of Computer and Information Science
University of Oregon

Abstract

Given generators for a permutation group G on n letters, the question has been posed as to whether or not a natural polynomial time Greedy Algorithm could be used to find a minimum base for G . We show that the Greedy Algorithm does not always find a minimum base for G . In fact, the minimum base problem is NP-hard, and the corresponding decision problem, "does there exist a base for G of size no more than L ?", is NP-complete.

Let G be a permutation group of degree n , and let $\mathcal{M}(G)$ denote the size of a minimum base for G . Then the Greedy Algorithm produces a base of size $O(\mathcal{M}(G) \log \log n)$, and this bound is sharp with respect to worst case performance. The maximum size of a nonredundant base for G is bounded above by $O(\mathcal{M}(G) \log n)$. This, too, is optimal in the sense that there exist an infinite number of permutation groups that realize this upper bound.

1 Introduction

The concepts of a base and strong generating set were introduced by Sims in 1970 and used as an efficient method to store and analyze large permutation groups [?], [?]. Since then bases and strong generating sets have played a key role in the development of numerous group theoretic algorithms. Furst, Hopcroft, and Luks showed that Sims' algorithm for computing a base and strong generating set runs in time $O(n^6)$ [?]. In 1982 Jerrum described an algorithm for computing a base and strong generating set with running time $O(n^5)$ [?]. Later we shall point out how a reduction in the size of the base can result in a reduction in both the time and space complexity of any algorithm that uses a base and strong generating set.

With this savings in mind a polynomial time Greedy Algorithm (described later) was suggested in [?, pages 10, 15] for constructing a small base for a permutation group G , specified by generators. The question then arose as to whether or not this Greedy Algorithm would always find a minimum base for G [?]. We show that this is not the case, and that the problem of finding a minimum base for G is, in fact, NP-hard. The corresponding decision problem of determining whether there exists a base of size less than or equal to L (a positive integer) is NP-complete. Moreover, both problems remain NP-hard even if we restrict G to be a cyclic group or an elementary abelian p-group.

*Research supported by ONR Grant N00014-86-0419.

Under the assumption that $P \neq NP$, we may assume that there does not exist a polynomial time algorithm that computes minimum bases for permutation groups. Thus, the most that one can hope to do efficiently is find minimum bases for special classes of groups, or find algorithms that will compute “small” bases (i.e., approximation algorithms). To determine if a base is small we shall compare its size to the size of a largest (nonredundant) base, and the size of a minimum base. The following results indicate that the Greedy Algorithm is a reasonable approximation algorithm.

Let G be a finite permutation group of degree n , then $O(\mathcal{M}(G) \log n)$ is an upper bound for the size of a nonredundant base for G . Moreover, for any $k \geq 1$ and for any n sufficiently large, there exists a permutation group G of degree n such that, $\mathcal{M}(G) = k$, and G has a nonredundant base of size at least $c(k \log n)$. Here, c is a constant that does not depend on k or n . Thus, we may conclude that the upper bound $O(\mathcal{M}(G) \log n)$ is “sharp”.

In comparison the size of a base produced by the Greedy Algorithm is bounded above by $O(\mathcal{M}(G) \log \log n)$. This bound is also “sharp”, since for any $k \geq 2$ and for any n sufficiently large, there exists a permutation group G of degree n with, $k = \mathcal{M}(G)$, and a base produced by the Greedy Algorithm of size at least $c'(\mathcal{M}(G) \log \log n)$. Once again c' is a constant that does not depend on k or n .

This paper is organized as follows. Definitions, basic mathematical tools, and a description of the Greedy Algorithm are found in section 2. In this section we also provide an example to show how the Greedy Algorithm fails to find a minimum base. Section 3 contains applications for a base and the motivation for finding a small base. In section 4 we prove that the minimum base problem is NP-hard. In section 5 we analyze the performance of the Greedy Algorithm. In the last section we summarize our results and describe several problems.

2 Preliminaries

2.1 Definitions

Let $\Omega = \{1, 2, \dots, n\}$ and let G be a subgroup of $\text{Sym}(\Omega)$ ($G \leq \text{Sym}(\Omega)$). We define the *product* of two permutations $g, h \in G$ on $\omega \in \Omega$ by the formula $\omega^{gh} = (\omega^g)^h$. By the *degree* of $G \neq \{1\}$ we mean the number of points moved by G .

Let $\omega \in \Omega$, then we call $\{\omega^g | g \in G\}$ the G -orbit of ω . 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 $G_A = G_a$. If $H \leq G$, then we define an equivalence relation on G in which two elements $g, h \in G$ are equivalent $\Leftrightarrow gh^{-1} \in H$. The equivalence classes of G under this relation are called the right cosets (Hg) of H in G , and we let $G : H$ denote the set of equivalence classes. By $|G|$ we denote the order of G and by $|G : H|$ the *index* $|G|/|H|$ of H in G .

For any abstract group G , we define $G^* \leq \text{Sym}(G)$, called the *right regular representation* of G . The elements of G^* are constructed as follows. For each g in G we define the permutation $g^* \in G^*$ which acts on G via right multiplication. That is, for each “point” x in G , $x^{g^*} = xg$.

The above definitions and notation are taken from [13]. The following definitions are due to Sims and may be found in [11].

A *base* for G is a sequence of points $B = b_1, b_2, \dots, b_k$, $b_i \in \Omega$, 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 $G = G^0 \geq G^1 \geq \dots \geq G^k = \{1\}$ where G^i is the

subgroup of G that fixes $b_1, \dots, b_i, 1 \leq i \leq k$ is the chain of stabilizers of G relative to B . We shall call a base *nonredundant* if each of the inclusions $G^{i-1} \geq G^i$ is proper. A *strong generating set* for G relative to B is a subset Z of G such that G^i is generated by $Z \cap G^i, 0 \leq i \leq k - 1$.

2.2 Problems, Running time, and Poly-time Reductions

The following is the list of problems that we are interested in.

- X3C** Input: A finite set Y with $|Y| = 3q$ and a collection M of 3-element subsets of Y .
 Question: Does M contain a subcollection M' such that every element of Y is contained in exactly one member of M' ?
- SB** Input: $G \leq \text{Sym}(\Omega)$ given by generators and a positive integer $L \leq |\Omega|$.
 Question: Does there exist a base for G of size no more than L ?
- MEM** Input: $G \leq \text{Sym}(\Omega)$ given by generators, and $g \in \text{Sym}(\Omega)$.
 Question: Is $g \in G$?
- MB** Input: $G \leq \text{Sym}(\Omega)$ given by generators.
 Output: A minimum base $B = b_1, b_2, \dots, b_k$ for G .
-
- ORD** Input: $G \leq \text{Sym}(\Omega)$ given by generators.
 Output: $|G|$.

The *problem size* of MB, SB, MEM, and ORD is a function of the degree of the group and the number of generators used to specify the group. The problem size of X3C is $3q + |M|$. We say that there is a polynomial time reduction from one problem to another if there is a polynomial time algorithm that transforms any instance of the first problem into an instance of the second problem, and there exist a solution to the first problem instance if and only if there exist a solution to the second. Thus, if we have a polynomial time algorithm for the second problem we can convert this to a polynomial time algorithm to the first.

In [7] it is shown that X3C is NP-complete. A discussion of the implications of this statement can be found in the first two chapters of [7]. In short, if we assume that $P \neq NP$ (a statement that many believe to be true), then there does not exist a polynomial time solution to X3C. In section 4 we reduce X3C to SB, which in turn suggests that there does not exist a polynomial time solution to SB or MB.

When discussing the running time and space complexity of algebraic algorithms we shall follow convention and assume the size of the input is linear in the degree of the group.

2.3 Mathematical Tools

The following facts shall be used throughout the paper and proofs of these statements may be found in either [13], or [10].

fact 1 Let $G \leq \text{Sym}(\Omega)$ and $\omega \in \Omega$. If r is the size of the G -orbit of ω , then $|G : G_\omega| = r$.

fact 2 Let G be a finite group and $g \in G$. Then $G^* \cong G$ (G^* isomorphic to G) and $(G^*)_g = \{1\}$.

fact 3 Let $G \leq \text{Sym}(\Omega)$, and let $A = a_1, a_2, \dots, a_k$ be a base for G . If $r_i = |G^{i-1} : G^i|$ for $1 \leq i \leq k$, then (by Lagrange's theorem) $|G| = r_1 r_2 \cdots r_k$.

Remark 2.1 Given $G = \langle \sigma \rangle \leq \text{Sym}(\Omega)$ and base $B = b_1, b_2, \dots, b_k$ 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 $r = \text{lcm}\{r_1, r_2, \dots, r_m\}$, $1 \leq m \leq k$.

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

The following notation will be used in section 4 and in section 5. Let X be any finite set and X_1, X_2, \dots, X_r be subsets of X such that their union is all of X . Let $K \cong (\mathcal{Z}_p)^{|X|}$ (p any fixed prime), and $K = \langle \sigma_x | x \in X \rangle$. For each X_i define $H_i \leq K$ by $H_i = \langle \sigma_x | x \in X_i \rangle$ ($H_i \cong (\mathcal{Z}_p)^{|X_i|}$).

Let Ω be the disjoint union of the H_i , and let $H = H_1^* \times H_2^* \times \dots \times H_r^*$. Now each element of H is an r -tuple (h_1, \dots, h_r) , and we may view H as a subgroup of $\text{Sym}(\Omega)$ as follows: if $\omega \in \Omega$, say $\omega \in H_i$, then $\omega^{(h_1, \dots, h_r)} = \omega h_i$.

Define a homomorphism $\Pi : K \rightarrow H$ via $\Pi(\sigma_x) = (h_1, \dots, h_r)$ where $h_i = \sigma_x^*$ if $x \in X_i$ and $h_i = 1$ otherwise. Note that $\ker(\Pi) = \{1\}$, since every element $x \in X$ is in some X_i . Thus, $K \cong \Pi(K)$, and we may view $\Pi(K)$ as a permutation group on Ω .

The reader may find it helpful to look at the example given at the end of this section, where the groups $K, \Pi(K)$, and H are constructed from sets $X = \{a, b, c, d, e, f\}$, $X_1 = \{a, b\}$, $X_2 = \{c, d\}$, $X_3 = \{e, f\}$, and $X_4 = \{a, c, e\}$.

Remark 2.2 Let $X' \subseteq X$, and $K' \leq K$ defined by $K' = \langle \sigma_x | x \in X' \rangle$. If $\omega \in H_i$, then $\Pi(K')_\omega = \Pi(\langle \sigma_x | x \in (X' \setminus X_i) \rangle)$.

Proof: $\Pi(K')_\omega = \{(h_1, \dots, h_r) \in \Pi(K') | h_i = 1\} = \Pi(\langle \sigma_x | x \in (X' \setminus X_i) \rangle)$

The first equality follows from fact 2 and the second follows from the definition of Π . \square

Remark 2.3 Let G be any finite group and $H \leq G$. Then there exists $H' \leq G^*$, with $H' \cong H$, and H' partitions the set G into $|G : H|$ H' -orbits of size $|H|$.

Proof: Let $\Phi : G \rightarrow G^*$ denote the isomorphism between G and G^* given by $\Phi(g) = g^*$. Then $\Phi(H) = H'$ and the left cosets (gH) of H in G are the H' -orbits. \square

2.4 The Greedy Algorithm

In this section we outline Jerrum's algorithm [8] for computing a base, and strong generating set. We then explain how the algorithm can be modified to include the Greedy heuristic. The input to Jerrum's algorithm is a group $G \leq \text{Sym}(\{1, 2, \dots, n\})$ specified by generators. The output of the algorithm is the base $B = 1, 2, \dots, n$ and a data structure that contains a strong generating set for G .

[Jerrum's Algorithm]

for $i = 1$ to n do begin

- (1) Using generators for G^{i-1} compute a set of coset representatives for G^i in G^{i-1} .
- (2) Update the data structure.
- (3) Compute a set of $O(n^2)$ Schreier generators for G^i .

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

end

In [1, pages 10, 15], it was observed that there was no need to specify the base in advance; point b_i can be chosen after a set of generators for G^{i-1} is known. The Greedy heuristic is to pick a point b_i that will force $|G^i|$ to be as small as possible. Since $|G^i| = \frac{|G^{i-1}|}{r}$, where r is the order of the G^{i-1} -orbit of b_i , this suggests that we choose b_i from a largest G^{i-1} -orbit.

[Jerrum's Algorithm with Greedy heuristic]

$i = 1$.

while $G^{i-1} \neq 1$ do begin

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

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

(2) Update the data structure.

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

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

(5) $i = i + 1$.

end while

end

The running time of this algorithm is dominated by step (4). Thus, Jerrum's algorithm 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 sets X and X_i are used to build groups K , $\Pi(K)$, and H described above. Second, it shows that the Greedy Algorithm fails to find a minimum base for the specific $\Pi(K)$ constructed.

example # 2: Let $X = \{a, b, c, d, e, f\}$, $X_1 = \{a, b\}$, $X_2 = \{c, d\}$, $X_3 = \{e, f\}$, and $X_4 = \{a, c, e\}$. Following the construction outlined above (with $p = 2$) we define the following 5 abstract groups: $K = \langle \sigma_x | x \in X \rangle$ ($K \cong (\mathbb{Z}_2)^6$), $H_1 = \langle \sigma_a, \sigma_b \rangle$, $H_2 = \langle \sigma_c, \sigma_d \rangle$, $H_3 = \langle \sigma_e, \sigma_f \rangle$, $H_4 = \langle \sigma_a, \sigma_c, \sigma_e \rangle$, and each $H_i \leq K$.

Then Ω is the disjoint union of the H_i , and $H = H_1^* \times H_2^* \times H_3^* \times H_4^*$. We write each element of H as a 4-tuple (h_1, \dots, h_4) , and view H as a subgroup of $\text{Sym}(\Omega)$ via $\omega \in \Omega$, then $\omega^{(h_1, \dots, h_4)} = \omega h_i$ if $\omega \in H_i$. The monomorphism $\Pi : K \rightarrow H$ maps the generators of K to the following elements in H : $\Pi(\sigma_a) = (\sigma_a^*, 1, 1, \sigma_a^*)$, $\Pi(\sigma_b) = (\sigma_b^*, 1, 1, 1)$, $\Pi(\sigma_c) = (1, \sigma_c^*, 1, \sigma_c^*)$, $\Pi(\sigma_d) = (1, \sigma_d^*, 1, 1)$, $\Pi(\sigma_e) = (1, 1, \sigma_e^*, \sigma_e^*)$, $\Pi(\sigma_f) = (1, 1, \sigma_f^*, 1)$.

Let $G = \Pi(K)$, 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 Ω , and $G = G^0 \geq G^1 \geq G^2 \geq G^3 \geq G^4$ be the chain of stabilizers of G relative to B . Then using remark 2.2 we compute generators for each group in the chain.

<i>Fixed Points</i>	<i>Group Generators</i>
<i>none</i>	$G^0 = \Pi(\langle \sigma_a, \sigma_b, \sigma_c, \sigma_d, \sigma_e, \sigma_f \rangle)$
$b_1 \in H_4$	$G^1 = \Pi(\langle \sigma_b, \sigma_d, \sigma_f \rangle)$
$b_2 \in H_1$	$G^2 = \Pi(\langle \sigma_d, \sigma_f \rangle)$
$b_3 \in H_2$	$G^3 = \Pi(\langle \sigma_f \rangle)$
$b_4 \in H_3$	$G^4 = \{1\}$

Since $G^4 = \{1\}$ we know that B is a base for G . Furthermore, since we know the generators for each group in the chain we can use remark 2.3 to find the orbit structure of each group in the chain.

<i>Group</i>	<i>Orbit</i>				<i>Structure</i>
G^0	H_1	H_2	H_3	H_4	
G^1	$\{1, \sigma_b\} \{ \sigma_a, \sigma_a \sigma_b \}$	$\{1, \sigma_d\} \{ \sigma_c, \sigma_c \sigma_d \}$	$\{1, \sigma_f\} \{ \sigma_e, \sigma_e \sigma_f \}$	<i>trivial</i>	
G^2	<i>trivial</i>	$\{1, \sigma_d\} \{ \sigma_c, \sigma_c \sigma_d \}$	$\{1, \sigma_f\} \{ \sigma_e, \sigma_e \sigma_f \}$	<i>trivial</i>	
G^3	<i>trivial</i>	<i>trivial</i>	$\{1, \sigma_f\} \{ \sigma_e, \sigma_e \sigma_f \}$	<i>trivial</i>	
G^4	<i>trivial</i>	<i>trivial</i>	<i>trivial</i>	<i>trivial</i>	

Any base for G produced by the Greedy Algorithm must begin with a point $b_1 \in H_4$, since H_4 is the largest G -orbit. The reader may verify that the base B is one that the Greedy Algorithm could produce. From the orbit structure of G^1 it is clear that the Greedy Algorithm will always produce a base for G of size 4. A minimum base for G has size 3 and consist of one point from each of the 3 sets H_1, H_2 , and H_3 .

3 Applications and Motivation

We mentioned in the introduction that the size of the base affects the space and time complexity of algebraic algorithms that use a base and strong generating set. In particular, it was shown in [2] that the running time of Jerrum's algorithm is $O(k^2 n^3)$, where k is the size of the base produced by Jerrum's algorithm and n is the degree of the group. Once one has a base and strong generating set it is quite easy to solve both MEM and ORD. In fact, MEM can be solved in $O(nk)$ time and ORD is the product of k integers [6].

A small base can also lead to a reduction in the space required to store elements from the group, since every element in the group is completely determined by its action on the base. Very small bases are also critical for CAYLEY [4], and for various computer constructions of simple groups.

Given generators for a group $G \leq \text{Sym}(\Omega)$ one is often interested in finding subgroups of G that have a particular properties (e.g., finding the center of G). One technique used to compute such groups is backtrack search [3]. A small base can be used to improve the time complexity of backtrack search algorithms, since the size of the base limits the depth of the backtrack search tree [9].

There is one final motivation for finding a small base that we should mention. In [1] the Greedy

Algorithm was used along with other heuristics to speed up the overall running time of their backtracking algorithm with symmetry.

4 Reductions

We shall show that even when the group G is restricted to cyclic groups or elementary abelian p -groups, the small base (SB) problem is NP-complete, which implies that minimum base (MB) is NP-hard.

It is not difficult to show that the SB problem is in NP. We guess a base for G , $B = b_1, b_2, \dots, b_k$, and check that $k \leq L$. Then use Jerrum's algorithm to verify that B is a base for G .

Theorem 4.1 *SB is NP-complete even if G is constrained to be a cyclic group.*

Proof: We have already shown that SB is in NP. Thus, it will suffice 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 $P = \{p_1, p_2, \dots, p_{3q}\}$ be the set of the first $3q$ primes. Define $f : Y \rightarrow P$ such that f is injective. For each 3-set $m_i = \{x, y, z\}$ in M let $s_i = f(x)f(y)f(z)$, $i = 1, \dots, r$.

Let $n = \sum_{i=1}^r s_i$, and let $\Omega = \{1, 2, \dots, n\}$. Construct $\sigma \in \text{Sym}(\Omega)$ with cycle decomposition consisting of r disjoint s_i -cycles C_i , $1 \leq i \leq r$. From this we construct an instance of SB, where $G = \langle \sigma \rangle$ and $L = q$. This is a polynomial time reduction since n is $O(q^3(\log q)^3 r)$, which is polynomial in $3q + r$. To finish the proof we need only show that our instance of X3C has a solution iff the instance of SB has a solution.

Let $B = b_1, b_2, \dots, b_k$ be a sequence of points from Ω , where b_i is a point in cycle $C_{i'}$, and $k \leq q$. Let $s = \text{lcm}\{s_{1'}, s_{2'}, \dots, s_{k'}\}$, then by remark 2.1 we have

$$\begin{aligned} B \text{ is a base for } G &\Leftrightarrow s = p_1 p_2 \cdots p_{3q} = |G| \\ &\Leftrightarrow k = q \text{ and the } s_i \text{ are all relatively prime} \\ &\Leftrightarrow \text{the sets } m_{1'}, m_{2'}, \dots, m_{q'} \text{ are disjoint} \\ &\Leftrightarrow \text{the } q \text{ 3-sets } m_{1'}, m_{2'}, \dots, m_{q'} \text{ in } M \text{ cover } Y. \quad \square \end{aligned}$$

Corollary 4.2 *Finding a minimum base for a cyclic group is NP-hard.*

Proof: This follows immediately from theorem 4.1. \square

Thus, the MB problem appears intractable even for cyclic groups. In the above reduction of X3C to SB the order 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 "small" orbits. The following theorem proves that the SB problem remains difficult even if we restrict the group to have orbits of order no more than 8.

Theorem 4.3 *SB is NP-complete even if we restrict G to be an elementary abelian p -group, with orbits of size no more than p^3 (p a prime).*

Proof: It will suffice to show that X3C reduces to SB where the group constructed for the SB problem is a elementary abelian p -group.

Let (Y, M) be an instance of X3C where $|Y| = 3q$, $M = \{m_1, m_p, \dots, m_r\}$. We shall assume, without loss of generality, that each y in Y is contained in at least one m_i .

Now we shall use the construction outlined in section 2.3 to define groups K , $\Pi(K)$, and H . Let $X = Y$, and $X_i = m_i$, $1 \leq i \leq r$. This gives us a group $K \cong (\mathcal{Z}_p)^{3q}$, a group $H \leq \text{Sym}(\Omega)$, and a monomorphism $\Pi : K \rightarrow H$.

Let $G = \Pi(K) = \langle \Pi(\sigma_x) | x \in X \rangle$, and $L = q$ be our instance of SB . This is a polynomial time reduction, since $|\Omega| = rp^3$ and there are only $3q$ generators. Hence, to finish the proof we need only show that our instance of SB has a solution iff the instance of X3C has a solution.

Let $B = b_1, b_2, \dots, b_k$ ($k \leq q$) be a sequence of points with b_i in G -orbit $H_{i'}$. Let $Y_0 = Y$, and let $Y_i = Y_{i-1} \setminus m_{i'}$ for $1 \leq i \leq k$. Then by remark 2.2 we have $G^i = \Pi(\langle \sigma_y | y \in Y_i \rangle)$, for $1 \leq i \leq k$.

B is a base for $G \Leftrightarrow G^k = \Pi(\langle \sigma_y | y \in Y_k \rangle) = 1$
 $\Leftrightarrow Y_k = \emptyset$
 $\Leftrightarrow k = q$, and all the $m_{i'}$ are disjoint
 $\Leftrightarrow m_{1'}, m_{2'}, \dots, m_{q'}$ cover X . \square

Corollary 4.4 *MB is NP-hard even if G is restricted to be an elementary abelian p -group.*

Proof: This follows immediately from theorem 4.3. \square

5 The Greedy Algorithm and Bounds for the Size of a Base

We know that the Greedy Algorithm cannot be used to solve MB. One might now ask whether the Greedy Algorithm is a good approximation heuristic for computing a small base. To answer this question we need to know (1) how big can a nonredundant base be, and (2) how big can a base produced by the Greedy Algorithm be? In this section we shall answer these two questions by comparing the sizes of these bases with size of a minimum base.

5.1 Bounds for the Size of a Nonredundant Base

Remark 5.1 *Let G be a permutation group of degree n , and suppose that all the orbits of G have order no more than c . Let r be the size of any nonredundant base for G , then $r \leq \mathcal{M}(G) \log c$ ¹.*

Proof: Fact 1 and fact 3 imply that $2^r \leq |G| \leq c^{\mathcal{M}(G)}$. \square

Lemma 5.2 *Let $G \leq \text{Sym}(\{1, 2, \dots, n\})$, and let r be the size of a nonredundant base for G . Then r is $O(\mathcal{M}(G) \log n)$ and this bound is sharp.*

Proof: By remark 5.1 we have $r \leq \mathcal{M}(G) \log n$. To prove that the bound is sharp it will suffice to show that for any $k \geq 1$ and for any n sufficiently large, there exists a permutation group G on n points such that,

¹throughout this paper $\log = \log_2$

1. $\mathcal{M}(G) = k$.
2. G has a nonredundant base of size $\Omega(k \log n)$.

Once again we shall use notation from section 2.3. Let $X = \{1, 2, \dots, rk\}$, where $r \geq 1 + \log k$. Let $X_i = \{r(i-1) + 1, \dots, ir\}$, $1 \leq i \leq k$, and $X_{j+k} = \{j\}$, $1 \leq j \leq rk$. Let $G = \Pi(K) \cong (\mathcal{Z}_p)^{rk}$. If we let $p = 2$, then $G \leq \text{Sym}(\Omega)$ and $|\Omega| = k2^r + 2rk = n$.

Since a largest G -orbit has size 2^r and $|G| = 2^{rk}$ it follows from facts 1 and 3 that a minimum base for G must have size at least k . Let $A = a_1, a_2, \dots, a_k$ where $a_i \in H_i$, then by remark 2.2 it follows that A is a minimum base for G .

Now we shall show that the group G has a nonredundant base of size at least $\frac{1}{2}k \log n$. Let $B = b_1, b_2, \dots, b_{rk}$ where $b_j \in H_{k+j}$, and let $G = G^0 \geq G^1 \geq \dots \geq G^{rk}$ be the chain of stabilizers of G relative to B . Remark 2.2 implies that $|G^i| = 2^{rk-i}$, and it follows that B is a nonredundant base for G , and the size of B is $rk \geq \frac{1}{2}k \log n$. Given k we have shown that for every integer $r \geq 1 + \log k$ there exists a permutation group of degree $n = k2^r + 2rk$ satisfying properties 1 and 2. The fact that $k2^r < n < k2^{r+1}$ is enough to insure that the theorem holds for all values of n sufficiently large. \square

5.2 Bounds for the Greedy Algorithm

Let G be a permutation group on n points and let $B = b_1, b_2, \dots, b_m$ be a base produced by the Greedy Algorithm. We shall show that m is bounded above by $O(\mathcal{M}(G) \log \log n)$. This upper bound is “sharp” with respect to worst case performance.

Remark 5.3 *If G be a permutation group of degree n , and $B = b_1, b_2, \dots, b_k$ is a base for G , then there exists a G -orbit of order $|G|^{\frac{1}{k}}$.*

Proof: Follows from fact 1 and fact 3. \square

Lemma 5.4 *Let $G \leq \text{Sym}(\{1, 2, \dots, n\})$ and $B = b_1, b_2, \dots, b_m$ a base for G produced by the Greedy Algorithm, then m is bounded above by $O(\mathcal{M}(G)(\log \log |G| - \log \mathcal{M}(G)) + \mathcal{M}(G))$.*

Proof: Let $G = G^0 \geq G^1 \geq \dots \geq G^m = \{1\}$ be the chain of stabilizers of G relative to B . First we prove by induction that,

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

The statement holds trivially when $i = 0$. Assuming the statement is true for i , we shall show that it is true for $i + 1$. Since $\mathcal{M}(G^i) \leq \mathcal{M}(G)$ it follows from remark 5.3 that G^i must have an orbit of order 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 order of a largest G^i -orbit. Thus $|G^i : G^{i+1}| \geq |G^i|^{\frac{1}{\mathcal{M}(G)}}$, and this implies that $|G^{i+1}| \leq |G^i|^{(\mathcal{M}(G)-1)/\mathcal{M}(G)} \leq |G|^{((\mathcal{M}(G)-1)/\mathcal{M}(G))^{i+1}}$.

$$i = \lceil 2\mathcal{M}(G)(\log \log |G| - \log \mathcal{M}(G)) \rceil \Rightarrow |G^i| \leq 2^{\mathcal{M}(G)}. \quad (2)$$

This statement is proved by the following implications:

$$i = \lceil 2\mathcal{M}(G)(\log \log |G| - \log \mathcal{M}(G)) \rceil$$

$$\begin{aligned}
&\Rightarrow i \geq 2\mathcal{M}(G)(\log \log |G| - \log \mathcal{M}(G)) \\
&\Rightarrow i \geq \frac{(\log \mathcal{M}(G) - \log \log |G|)}{\log((\mathcal{M}(G)-1)/\mathcal{M}(G))} \\
&\Rightarrow i \log((\mathcal{M}(G)-1)/\mathcal{M}(G)) + \log \log |G| \leq \log \mathcal{M}(G) \\
&\Rightarrow ((\mathcal{M}(G)-1)/\mathcal{M}(G))^i \log |G| \leq \mathcal{M}(G) \\
&\Rightarrow |G|^{((\mathcal{M}(G)-1)/\mathcal{M}(G))^i} \leq 2^{\mathcal{M}(G)} \\
&\Rightarrow |G|^i \leq 2^{\mathcal{M}(G)} \text{ by 1.}
\end{aligned}$$

To finish the proof of the lemma first observe that if a group has order less than or equal to $2^{\mathcal{M}(G)}$ then any nonredundant base for that group has size less than or equal to $\mathcal{M}(G)$. Now, combining 2 with the fact that the Greedy Algorithm produces a nonredundant base, we are done. \square

Theorem 5.5 *Let G be a permutation group of degree n and let $B = b_1, b_2, \dots, b_m$ be a base for G produced by the Greedy Algorithm. Then m is bounded above by $O(\mathcal{M}(G) \log \log n)$.*

Proof: By remark 5.3 it follows that $n \geq |G|^{\mathcal{M}(G)}$. But this implies that $\mathcal{M}(G) \log n \geq \log |G|$. From lemma 5.4 we know that there exists a constant c such that,

$$\begin{aligned}
m &\leq c(\mathcal{M}(G)(\log \log |G| - \log \mathcal{M}(G)) + \mathcal{M}(G)) \\
&\leq c(\mathcal{M}(G)(\log(\mathcal{M}(G) \log n) - \log \mathcal{M}(G)) + \mathcal{M}(G)) \\
&\leq c(\mathcal{M}(G) \log \log n + \mathcal{M}(G)).
\end{aligned}$$

Therefore, for $n \geq 4$ there exists constant c' such that, $m \leq c' \mathcal{M}(G) \log \log n$. \square .

We shall conclude this section with a theorem that proves that the bound given in theorem 5.5 is “sharp”. The following technical lemma will be needed.

Lemma 5.6 *Let r, k be to positive integers such that $r \geq k^2 \geq 4$. Define $r_0 = r$ and $r_i = r_{i-1} - \lceil r_{i-1}/k \rceil$ 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 $s_i \leq r_i \forall i$. Let $\alpha = (1 - 1/k)$, then $s_i = \alpha^i s_0 - \frac{1-\alpha^i}{1-\alpha}$. Thus

$$\begin{aligned}
s_i \geq 1 &\Leftrightarrow \alpha^i s_0 - \frac{1-\alpha^i}{1-\alpha} \geq 1 \\
&\Leftrightarrow \alpha^i (s_0 + k) \geq k + 1 \\
&\Leftrightarrow i \log \alpha + \log(s_0 + k) \geq \log(k + 1) \\
&\Leftrightarrow i \leq \frac{\log(r+k) - \log(k+1)}{\log(k/k-1)}. \square
\end{aligned}$$

Theorem 5.7 *For any integer $k \geq 2$, and n sufficiently large there exists a permutation group G of degree n such that,*

1. $\mathcal{M}(G) = k$.
2. G has a base produced by the Greedy Algorithm of size $\Omega(\mathcal{M}(G) \log \log n)$.

Proof: Let Y be a set of order rk ($r \geq k^2$). The set is then partitioned into k sets of order r , $A_{1,0}, A_{2,0}, \dots, A_{k,0}$. 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 $\lceil \frac{|A_{i,j-1}|}{k} \rceil$ elements from $A_{i,j-1}$. The elements removed from the k sets $A_{1,j-1}, A_{2,j-1}, \dots, A_{k,j-1}$ are placed in a set Y_j . Note that $|Y_j| \geq |A_{i,j-1}|$. Let $\gamma = \lfloor (k/2)(\log(r+1) - \log k + 1) \rfloor$. The value of γ was computed in lemma 5.6 to insure that $|A_{i,j}| \geq 1$ for all values of i and j .

We shall use the sets defined above to construct elementary abelian 2-groups, K , $\Pi(K)$, and H , exactly as outlined in section 2 of this paper. Let $X = Y$, $X_i = A_{i,0}$ for $1 \leq i \leq k$, and $X_{k+j} = Y_j$ for $1 \leq j \leq \gamma$.

Let $G = \Pi(K)$, then G is a permutation group of degree $n = k2^r + 2^{|Y_1|} + 2^{|Y_2|} + \dots + 2^{|Y_\gamma|}$, and $G \cong (\mathcal{Z}_2)^{rk}$. To finish the proof we need to show that G satisfies the properties 1 and 2 stated above.

Consider the sequence of points $A = a_1, a_2, \dots, a_k$ where $a_i \in H_i$. Since $X = \dot{\bigcup}_{i=1}^k A_{i,0}$, it follows from remark 2.2 that A is a base for G . Moreover, since the largest G -orbit has size 2^r , facts 1 and 3 imply that A must be a minimum base for G .

Next we show that the Greedy Algorithm could select $B = b_1, b_2, \dots, b_\gamma$ as a partial base for G , where $b_j \in H_{j+k}$. It will suffice to show that b_j is in a G^{j-1} -orbit of maximal size. By remark 2.2 we have $G^{j-1} = \Pi(\langle \sigma_x | x \in (X \setminus \dot{\bigcup}_{l=1}^{j-1} X_{l+k}) \rangle) = \Pi(\langle \sigma_x | x \in \dot{\bigcup}_{i=1}^k A_{i,j-1} \rangle)$

Now using remark 2.3 we see that the points in H_i 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 H_{i+k} 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 $2^{|Y_j|}$, and this is a maximal G^{j-1} -orbit.

So far we have shown that G has a minimum base of size k , and that the Greedy Algorithm can produce a base of size at least γ . Since $r \geq k^2$ there exists a constant c' such that $\gamma \geq c'k \log r = c' \mathcal{M}(G) \log r$. To show that the (partial) base B has size $\Omega(\mathcal{M}(G) \log \log n)$ it will suffice to show that for some constant c'' , $r \geq c'' \log n$, where n is the degree of G .

$$\begin{aligned} n &= k2^r + 2^{|Y_1|} + 2^{|Y_2|} + \dots + 2^{|Y_\gamma|} \\ &\leq k2^r + 2^{r+k} + 2^{|Y_2|} + \dots + 2^{|Y_\gamma|} \\ &\leq k2^r + 2^{r+k+1}. \end{aligned}$$

Given k we have shown that for every integer $r \geq k^2$ there exists a permutation group of degree $n = k2^r + 2^{|Y_1|} + 2^{|Y_2|} + \dots + 2^{|Y_\gamma|}$ satisfying properties (1) and (2). The fact that $(k+1)2^r < n < (k+1)2^{r+1}$ is enough to insure that the theorem holds for all values of n sufficiently large. \square

6 Comments and Problems

In section 4 we showed that the MB problem was NP-hard. Under the assumption that $P \neq NP$ this implies that there does not exist a polynomial time algorithm for computing a minimum bases of permutation groups specified by generators. There are two common approaches to working on problems that are NP-hard. First, one can try to determine instances of the problem that are in P and then develop efficient algorithms for solving those problems. Theorems 4.1 and 4.3 suggest that the quite strong restrictions may be needed.

The second approach is to settle for approximations to the optimal solution. In section 5 it was shown that the Greedy Algorithm is a reasonable approximation algorithm. We close the paper with two problems.

Problem 1 Find a polynomial time algorithm that will solve MB for some restricted class of groups.
Problem 2 Find a polynomial time algorithm that computes smaller bases (worst case performance) than the Greedy Algorithm.

Acknowledgements

I would like to thank William Kantor for his helpful suggestions, and Eugene Luks for his ideas, interest and encouragement.

References

- [1] Brown, C., Finkelstein, L., Purdom, P. *Symmetry and Searching*, pre-print.
- [2] Brown, C., Finkelstein, L., Purdom, P. *Efficient Implementation of Jerrum's Algorithm for Permutation Groups*, pre-print.
- [3] Butler G. *Fundamental Algorithms for Permutation Groups*, Notes from lecture course on Symbolic and Algebraic Computation, 1984, 5/1-5/28.
- [4] Cannon, J. *Computational Toolkit For Finite Permutation Groups*, Proc. of Rutgers Group Theory, 1983-1984, eds. M. Aschbacher et al., Cambridge Univ. Press, 1984.
- [5] Finkelstein, L. Personal correspondence to E.M. Luks.
- [6] Furst, M., Hopcroft, J. and Luks, E.M. *Polynomial-time algorithms for permutation groups*, Proc. 21st IEEE FOCS, 1980, 36-41.
- [7] Garey, M., and Johnson, D. *Computers and Intractability: A guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.
- [8] Jerrum, M.R. *A Compact Representation for Permutation Group*, Proc. 23rd IEEE FOCS, 1982, 126-133.
- [9] Leon, J. *Computing Automorphism Groups of Combinatorial Objects*, Computational Group Theory, ed. M. Atkinson, Academic Press, 1984, 321-336.
- [10] Rotman, J. *The Theory of Groups*, Second Edition, Allyn and Bacon, 1979.
- [11] Sims, C.C. *Determining the Conjugacy Classes of a Permutation Group*, Computers in Algebra and Number Theory, eds. G. Birkhoff and M. Hall, Jr., SIAM-AMS Proc., Vol. 4, Amer. Math. Soc., 1970, 191-195.
- [12] Sims, C.C. *Computational methods in the study of permutation groups*, Computational Problems in Abstract Algebra, ed. J. Leech, Pergamon Press, 1970, 169-183.
- [13] Wielandt, H. *Finite Permutation Groups*, Academic Press. New York-London, 1964.