Chapter 11. Backtrack Search

In this chapter we consider searching the elements of a permutation group G with a base and strong generating set. The search aims to find a base and strong generating set for the subgroup of elements which satisfy a given property P. The search is through the tree of all base images of elements of G. Heuristic methods prune this tree. Some heuristics are *problem-dependent,* but others are applicable to all properties P, and are called *problem-independent.*

The backtrack search will enable us to construct a base and strong generating set for centralizers, normalizers, intersections, set stabilisers, and the automorphism group of a group.

Strong Generators of the Subgroup

Let P be a property of (some of) the elements of G. We require that P be decidable, so that given an element $g \in G$ it can be determined whether g satisfies property P or not. Furthermore, we require that the elements of G that satisfy the property P form a subgroup of G. We call this subgroup $H(P)$.

$$H(P) = \{ g \in G \mid g \text{ satisfies } P \}.$$

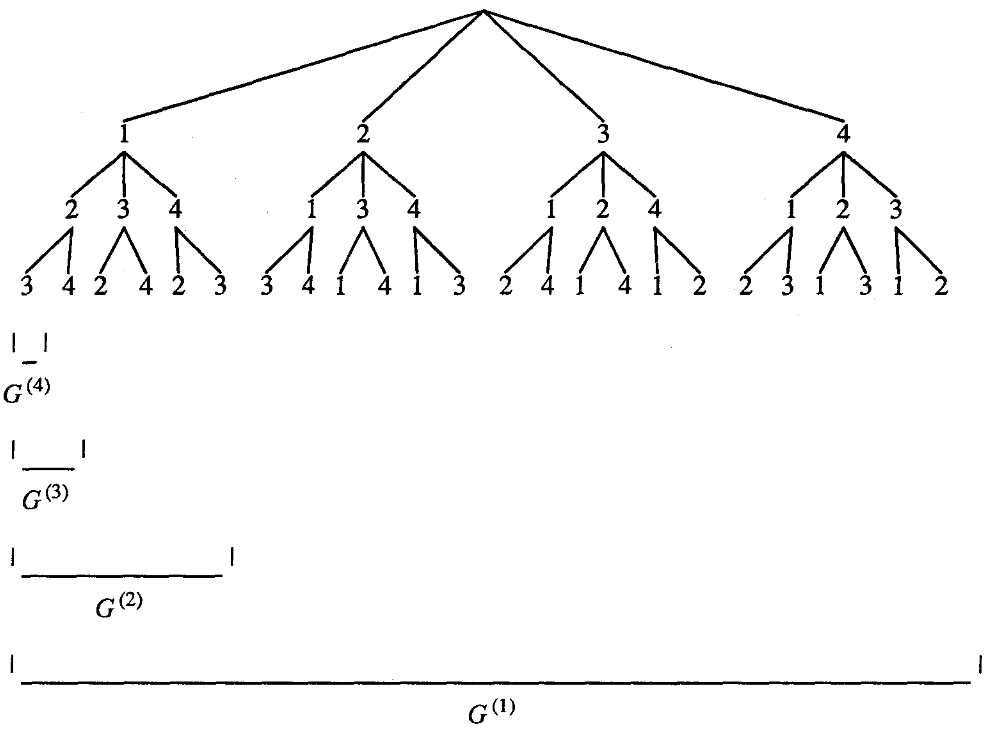
The first (and most important) consideration is that the algorithm construct a base and strong generating set of the subgroup, since they provide enough information to further investigate the subgroup. Any base for the group G will be a base for the subgroup $H(P)$, so the aim is to construct a strong generating set relative to the base for G. That is, we need generators for $H(P)^{(k)}, H(P)^{(k-1)}, \ldots, H(P)^{(2)}, H(P)^{(1)}$.

The algorithms for enumerating all elements (or all base images) will enumerate the elements of $G^{(k)}$ first, then the elements of $G^{(k-1)} - G^{(k)}, \ldots, G^{(1)} - G^{(2)}$, provided that the identity element is the "first" element of the set of coset representatives $U^{(i)}$, for all i. Modifying these enumerations to be searches for elements satisfying the property P, will lead naturally to the construction of $H(P)^{(k)}, \ldots, H(P)^{(1)}$, and hence to a strong generating set of $H(P)$.

It is best to think of these searches as enumerations of all base images, because then the search tree can be pruned using points, sets of points, etc rather than using elements, sets of elements, etc. Points are simply easier to manipulate than elements.

Figure 1 shows the tree of base images $[\gamma_1, \gamma_2, \ldots, \gamma_k]$ for the symmetric group of degree 4. The enumeration traverses the tree in preorder. This is also called a depth-first or backtrack traversal, hence the term, *backtrack search.*

Figure 1 : Symmetric Group of Degree 4



First in Orbit

As in the case of searching small groups, we can use cosets to prune the search of very large permutation groups. This is a problem-independent improvement. The information about cosets is translated into information about orbits, thus giving the first in the orbit criterion. To perform this translation, we must have a total order on the elements of the group.

Suppose we order the points of Ω, so that the base points $\beta_1, \beta_2, ..., \beta_k$ are the first k points of Ω. If we order the base images B^g in their lexicographical order, as follows,

$$B^g < B^{g'} \text{ if } [\beta_1^g, \beta_2^g, ..., \beta_i^g] = [\beta_1^{g'}, \beta_2^{g'}, ..., \beta_i^{g'}] \textbf{ and } \beta_{i+1}^g < \beta_{i+1}^{g'} \qquad (*)$$

and if we order the permutations g of G by their lexicographical order

$g < g'$ if g and g' have same images on the first i points of Ω
and the image of $i+1$st point under g
$<$ the image of $i+1$st point under g'

then these two orders actually coincide. That is,

$$g < g' \text{ if and only if } B^g < B^{g'}.$$

Let K be a subgroup of G. The elements of the right coset $K \underset{K}{\times} g$ have base images $B^{K \times g}$ and the elements of the left coset $\underset{K}{g} \times K$ have base images $\left[B^g \right]$. In particular, if $\beta_1{}^g$ is *not* the first point in the orbit $\left[\beta_1{}^g \right]$ of K, then g is *not* the first element of the left coset $g \times K$.

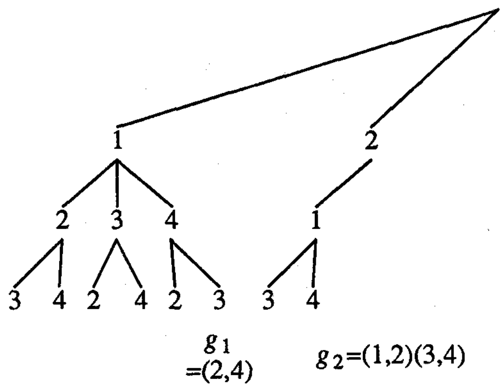
This generalizes to say that $\beta_i{}^g$ must be the first point in its K-orbit when searching for elements $g \in G^{(i)}$ where K is a known subgroup of $G^{(i)}$. Why? Well, when searching $G^{(i)} - G^{(i+1)}$, we know a subgroup K between $H(P)^{(i+1)}$ and $H(P)^{(i)}$. This is the subgroup of elements satisfying property P that we have already found. Another way of viewing the discarding of cosets as used in searching small groups is to say we only have to test the *first* element of each coset. That is, when searching $G^{(i)} - G^{(i+1)}$ we need only consider the elements g that are first in $g \times K$. In terms of base images this says that γ_i must be the first point in its orbit of K (since γ_i *is* $\beta_i{}^g$, for all elements g below the node $[\gamma_1, \gamma_2, \ldots, \gamma_i]$).

Let us see the effect of this in searching the symmetric group G of degree 4 for the elements that centralize z=(1,3)(2,4).

Initially, $K=\{id\}$, so its orbits are irrelevant. Searching $G^{(3)}$ we find nothing. Searching $G^{(2)}$ we find the base image [1,4,3] and the corresponding element g_1=(2,4). Thus $K=<g_1>$ with orbits {1} {2,4} {3}. Searching $G^{(1)}$ we must consider γ_1=2, and we find the element g_2=(1,2)(3,4) with base image [2,1,4]. Thus $K=<g_1, g_2>$ with orbits {1,2,3,4}. Since γ_1=2 is no longer first in its orbit, we no longer search the subtree with γ_1=2. The choices γ_1=3 or γ_1=4 also do not give elements first in the coset, so we are finished.

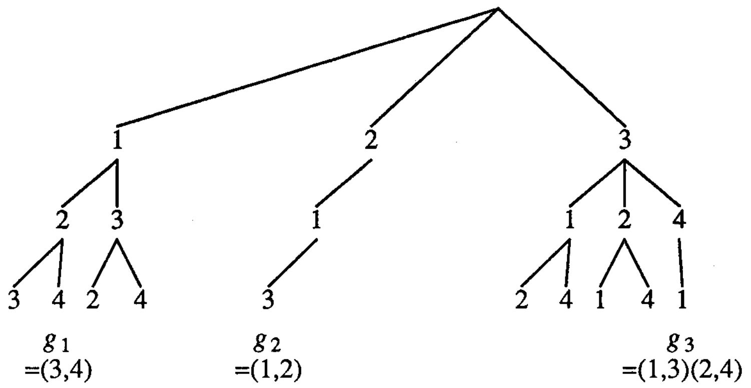
The tree we have searched is given in Figure 2.

Figure 2 : Centralizer of (1,3)(2,4) using First in Orbit



The tree we would have searched for the centralizer of z=(1,2)(3,4) is given in Figure 3.

Figure 3 : Centralizer of (1,2)(3,4) using First in Orbit



The search algorithm is presented in Algorithm 1 using two recursive procedures *search* and *generate*. In *generate* we wish to return through all the recursive levels of *generate* to procedure *search* once we have found a new element satisfying P. We do not strictly follow Pascal and include code to terminate each of the levels of recursion. Instead, we just indicate that this is what we wish to occur.

Algorithm 1 : Backtrack Search using First in Orbit

```
Input : a group $G$ with a base $B$=[$\beta_1$, $\beta_2$, ..., $\beta_k$] and a strong generating set $S$;
    a decidable property $P$ whose elements form a subgroup;

Output : a strong generating set relative to $B$ for the subgroup $K = H(P)$;

procedure search( $G$ : group; $P$ : property; $s$ : 1..$k$+1; var $K$ : group );
(* Search $G^{(s)}$ for the subgroup $K$ of elements satisfying $P$. *)
begin
  if $s = k+1$ then
    $K$ := {$id$};
  else
    search( $G, P, s+1, K$ );

    (*now search $G^{(s)} - G^{(s+1)}$ *)
    for each point $\gamma_s \in \Delta^{(s)}$ do
      if $\gamma_s$ is first in its $K$-orbit then
        generate( $G, P, s, s+1$, [$\beta_1$, $\beta_2$, ..., $\beta_{s-1}$, $\gamma_s$], $K$ );
      end if;
    end for;
  end if;
end;
```

```
procedure generate( G : group; P : property; s : 1..k+1; i : 1..k+1;
                    [$\gamma_1, \gamma_2, ..., \gamma_{i-1}$] : initial segment of base image;
                    var K : group );
(*
  Generate the elements of $G^{(s)}$ whose base image start
  with [$\gamma_1, \gamma_2, ..., \gamma_{i-1}$] and that may have property P.
  If one is found then extend K, the subgroup of $G^{(s)}$ of elements
  with property P that have already been found, and return to search.
*)
begin
  find an element $g \in G$ mapping [$\beta_1, \beta_2, ..., \beta_{i-1}$] to [$\gamma_1, \gamma_2, ..., \gamma_{i-1}$];
  if $i = k+1$ then
    if g satisfies P then
      K := <K, g>;
      return to search at level s;  (*since $\gamma_s$ is no longer first in its K-orbit*)
    end if;
  else
    for each point $\gamma_i$ in $\left[ \Delta^{(i)} \right]^g$ do
      generate( G, P, s, i+1, [$\gamma_1, \gamma_2, ..., \gamma_i$], K );
    end for;
  end if;
end;

begin
  search( G, P, 1, K );
end;
```

Restrictions on Image Points

This section looks at using the property P to prune the search. We are concerned with finding restrictions on the base images of elements that satisfy the property. Ideally, these restrictions should lead to a very restrictive test on the points γ_i of the base image, and the test should be quick and easy to compute.

We begin by looking at an example. Consider the property

$$P_z \; : \; g \in G \;\; \text{centralizes} \;\; z \in G$$

defining the centralizer of z in G. Recall that a consequence of g centralizing z is that, for any point β,

$$\beta^{g \times z} = \beta^{z \times g}.$$

Suppose we apply this fact to the case where $\beta_i = \beta_j^{\,z}$. Then

$$\gamma_i = \beta_i^{\,g} = (\beta_j^{\,g})^z = \gamma_j^{\,z}.$$

If $j < i$ then we know γ_j when we come to choose γ_i, so the choice for γ_i is restricted to at most one point, namely $\gamma_j^{\,z}$. Furthermore, suppose that $\beta_j^{\,z^r} = \beta_j$. Then $\gamma_j^{\,z^r} = \gamma_j$. Hence, the image γ_j of β_j is restricted to be a point in a z-cycle of the same length as the z-cycle containing β_j.

Figure 4 shows the result of applying this restriction to the construction of the centralizer of z=(1,3)(2,4) in the symmetric group of degree 4: the base is [1,2,3] so $\beta_3 = \beta_1{}^z$ which implies that $\gamma_3 = \gamma_1{}^z$. Figure 5 does the same for the centralizer of z=(1,2)(3,4), where $\beta_2 = \beta_1{}^z$.

Figure 4 : Centralizer of (1,3)(2,4) Restricting Choice of Image Points

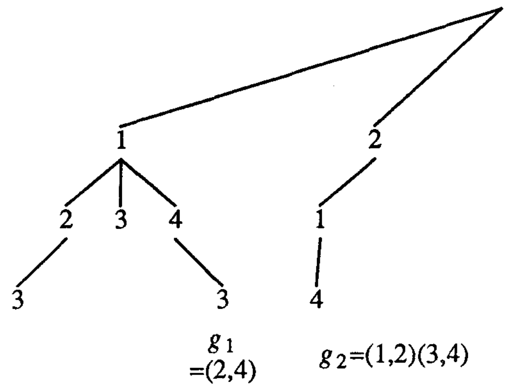
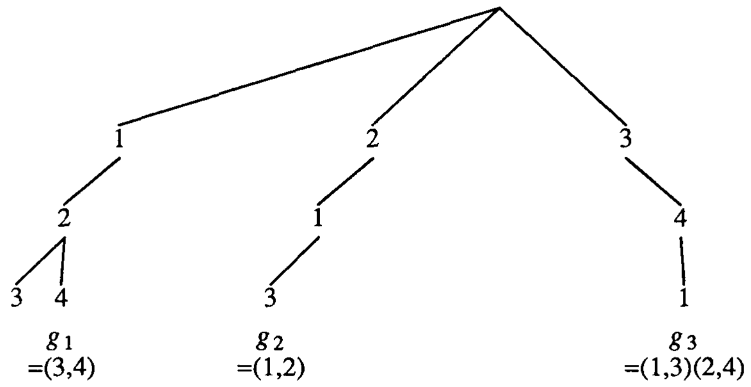


Figure 5 : Centralizer of (1,2)(3,4) Restricting Choice of Image Points



To incorporate such restrictions into the backtrack search we require information about the set

$$\Omega_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}]) = \{ \gamma_i \mid \text{there is an element of } G \text{ satisfying property } P \text{ whose base image begins with } [\gamma_1, \gamma_2, ..., \gamma_i] \}.$$

In general, we will not know this set exactly. One fact we do know is that, for any element $g \in G$ mapping $[\beta_1, \beta_2, ..., \beta_{i-1}]$ to $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$,

$$\Omega_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}]) \subseteq \left[\Delta^{(i)} \right]^g .$$

This is used as an upper bound on the set and its approximations. It will suffice for the backtrack search if we have an approximation

$$\overline{\Omega}_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}]) = \{ \gamma_i \mid \text{there may be an element of } G \text{ satisfying property } P \text{ whose base image begins with } [\gamma_1, \gamma_2, ..., \gamma_i] \}$$

which contains $\Omega_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}])$. The approximation is chosen to be as restrictive as possible, while still remaining easily computable.

For the centralizer problem, where $\beta_i = \beta_j^z$, $j < i$, we take

$$\overline{\Omega}_{P_z}([\gamma_1, \gamma_2, ..., \gamma_{i-1}]) = \{ \gamma_j^z \} \cap \left[\Delta^{(i)} \right]^g ,$$

and if β_i is not the image of some previous base point then we take

$$\overline{\Omega}_{P_z}([\gamma_1, \gamma_2, ..., \gamma_{i-1}]) = \{ \gamma \mid \text{the } z\text{-cycles of } \gamma \text{ and } \beta_i \text{ have the same length} \}$$

$$\cap \left[\Delta^{(i)} \right]^g .$$

Algorithm 2 presents the modifications that incorporate the use of the approximation $\overline{\Omega}_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}])$ to restrict the choice of base images. The procedure *search* has replaced the reference to $\Delta^{(s)}$ by $\overline{\Omega}_P([\beta_1, \beta_2, ..., \beta_{i-1}])$; and the procedure *generate* has replaced the reference to $\left[\Delta^{(i)} \right]^g$ by $\overline{\Omega}_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}])$.

Algorithm 2 : Backtrack Search Restricting Choice of Base Images

```
Input : a group $G$ with a base $B=[\beta_1, \beta_2, ..., \beta_k]$ and a strong generating set $S$;
    a decidable property $P$ whose elements form a subgroup;

Output : a strong generating set relative to $B$ for the subgroup $K = H(P)$;

procedure search( $G$ : group; $P$ : property; $s$ : $1..k+1$; var $K$ : group );
(* Search $G^{(s)}$ for the subgroup $K$ of elements satisfying $P$. *)
begin
  if $s = k+1$ then
    $K := \{id\}$;
  else
    search( $G, P, s+1, K$ );

    (*now search $G^{(s)} - G^{(s+1)}$ *)
    for each point $\gamma_s \in \overline{\Omega}_P(\beta_1, \beta_2, ..., \beta_{s-1})$ do
      if $\gamma_s$ is first in its $K$-orbit then
        generate( $G, P, s, s+1, [\beta_1, \beta_2, ..., \beta_{s-1}, \gamma_s], K$ );
      end if;
    end for;
  end if;
end;
```

```
procedure generate( $G$ : group; $P$ : property; $s$ : 1..$k$+1; $i$ : 1..$k$+1;
                    [$\gamma_1$, $\gamma_2$, ..., $\gamma_{i-1}$] : initial segment of base image;
                    var $K$ : group );
(* Generate the elements of $G^{(s)}$ whose base image start
   with $\gamma_1$, $\gamma_2$, ..., $\gamma_{i-1}$ and that may have property $P$.
   If one is found then extend $K$, the subgroup of $G^{(s)}$ of elements
   with property $P$ that have already been found, and return to search. *)
begin
  find an element $g \in G$ mapping [$\beta_1$, $\beta_2$, ..., $\beta_{i-1}$] to [$\gamma_1$, $\gamma_2$, ..., $\gamma_{i-1}$];
  if $i = k+1$ then
    if $g$ satisfies $P$ then
      $K := \langle K, g \rangle$;
      return to search at level $s$;  (*since $\gamma_s$ is no longer first in its $K$-orbit*)
    end if;
  else
    for each point $\gamma_i$ in $\bar{\Omega}_P$( [$\gamma_1$, $\gamma_2$, ..., $\gamma_{i-1}$] ) do
      generate( $G$, $P$, $s$, $i$+1, [$\gamma_1$, $\gamma_2$, ..., $\gamma_i$], $K$ );
    end for;
  end if;
end;

begin
  search( $G$, $P$, 1, $K$ );
end;
```

Choosing an Appropriate Base

By themselves, the restrictions on the images of base points are often not powerful enough. The most effective restrictions, like the centralizer restriction, use some relationship between the base points. However, the base that is given for the group may not display any of the required relationships between its points. So the development of effective backtrack searches involves an interplay between devising restrictions on the images of the base and devising ways of choosing a base that has the required relationships amongst its points in order for the restrictions to be effective.

Consider the example where G is the Mathieu group of degree 11. The group has a base [11,10,1,2]. For computing the centralizer of z=(2,5,3,9)(4,8,7,6), this base has no two base points β_i and β_j where $\beta_i = \beta_j^z$. Hence, the restrictions on base images will not be as effective as it might be. However, choosing the base [2,5,3,9] means that $\beta_i = \beta_{i-1}^z$, for i=2,3,4. This is the best we could have hoped for. Now choosing γ_1 uniquely determines the remainder of the base image of a centralizing element. The fact that the image of β_1=2 must lie in a z-cycle of length 4, and the first in the orbit test shows that the only values of γ_1 considered are 3 (leading to z^2 being found), and 4 (leading to (1,11)(2,4,9,6,3,7,5,8) being found).

Thus a group of order 7,920 has been searched by considering only 2 elements!

In general, the base appropriate to a problem is the one that minimises the size of the sets $\overline{\Omega}_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}])$ that are considered. Therefore, the information needed to compute the sets is collected before the search commences, and is used to guide the selection of an appropriate base.

For the centralizer problem, we wish to have $\beta_i = \beta_{i-1}{}^z$ as often as possible, so we choose the first base points to be in a cycle of the longest length. If the points of that cycle do not form a base then we choose, from amongst the remaining cycles, one of the longest length, and so on, until we have a base.

Once we have selected a base, the base change algorithm, described in the next chapter, obtains a strong generating set relative to the chosen base.

Using a Known Subgroup

This section looks at using the information that is often provided free with the problem description. For example, in computing the centralizer of z in G, we have not used the fact that z is in the centralizer. The search finds z, or elements from which we can form z. Another example is the construction of normalizers. The normalizer $N_G(H)$ of H in G contains the subgroup H, so the elements of H do not have to be found by searching.

Let L be the subgroup of G we know to be in $H(P)$. The first thing we do when searching $G^{(s)} - G^{(s+1)}$ is to add $L^{(s)}$ to the subgroup K we are constructing. Generally, we will know a strong generating set T of L, so this amounts to including the elements of $T^{(s)} - T^{(s+1)}$ in the strong generators of K. If we do not know a strong generating set of L then we just include whatever elements of $L^{(s)}$ we can easily lay our hands on. For example, the generators of L which fix the first $s-1$ base points.

The modifications to the backtrack search only affect the procedure *search*. These changes are presented in Algorithm 3.

Algorithm 3 : Backtrack Search Using Known Subgroup

Input : a group G with a base $B=[\beta_1, \beta_2, ..., \beta_k]$ and a strong generating set S;
a decidable property P whose elements form a subgroup;
a subgroup L of G whose elements satisfy P;
Output : a strong generating set relative to B for the subgroup $K = H(P)$;

```
procedure search( G : group;  P : property; s : 1..k+1; var K : group );
(* Search $G^{(s)}$ for the subgroup K of elements satisfying P.  *)
begin
  if $s = k+1$ then
    K := {id};
  else
    search( G, P, s+1, K );
    add $L^{(s)}$ to K; (*use known subgroup in search of $G^{(s)} - G^{(s+1)}$ *)
    for each point $\gamma_s \in \overline{\Omega}_P( [\beta_1, \beta_2, ..., \beta_{s-1}] )$ do
      if $\gamma_s$ is first in its K-orbit then
        generate( G, P, s, s+1, $[\beta_1, \beta_2, ..., \beta_{s-1}, \gamma_s]$, K );
      end if;
    end for;
  end if;
end;
```

Searching Images of an Initial Base Segment

A special case of a known subgroup is a stabiliser $G^{(l)}$ of the stabiliser chain. This leads to even more improvement than given in the last section because now we do not need to generate a complete base image but instead only the images of the first $l-1$ base points. The reason for this is that each such image determines a coset of $G^{(l)}$. We have $G^{(l)}$ in the subgroup K, so we need only test one element from each coset. There is no need to generate all the elements of that coset by generating the complete base images which extend the image of the initial base segment.

An example of this phenomenom is constructing the centralizer of z=(1,2) in the symmetric group of degree 4. The permutations which fix 1 and 2 must centralize z, so $G^{(3)}$ is a known subgroup of $H(P)$.

Another common example is the construction of set stabilisers. Here the group that stabilises all the points in the set individually is arranged to be a member of the stabiliser chain. It is a known subgroup of $H(P)$, and only the images of the points in the set have to be generated rather than the complete base image.

The modifications to Algorithm 2 are presented in Algorithm 4.

Algorithm 4 : Backtrack Search Using Shortened Base Images

Input : a group G with a base $B=[\beta_1, \beta_2, ..., \beta_k]$ and a strong generating set S;
a decidable property P whose elements form a subgroup;
Output : a strong generating set relative to B for the subgroup $K = H(P)$;

```
procedure search( G : group;  P : property; s : 1..k+1; l : 1..k+1; var K : group );
(* Search $G^{(s)}$ for the subgroup K of elements satisfying P.
  It is known that the elements of $G^{(l)}$ satisfy P.  *)
begin
  if $s = l$ then
    $K := G^{(l)}$;
  else
    search( G, P, s+1, l, K );
    (*now search $G^{(s)} - G^{(s+1)}$ *)
    for each point $\gamma_s \in \overline{\Omega}_P( [\beta_1, \beta_2, ..., \beta_{s-1}] )$ do
      if $\gamma_s$ is first in its K-orbit then
        generate( G, P, s, l, s+1, $[\beta_1, \beta_2, ..., \beta_{s-1}, \gamma_s]$, K );
      end if;
    end for;
  end if;
end;

procedure generate( G : group;  P : property; s : 1..k+1; l : 1..k+1; i : 1..k+1;
                    $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$ : initial segment of base image;
                    var K : group );
(* Generate the coset representatives of $G^{(l)}$ in $G^{(s)}$ whose base image start
  with $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$ and that may have property P.
  If one is found then extend K, the subgroup of $G^{(s)}$ of elements
  with property P that have already been found, and return to search.  *)
begin
  find an element $g \in G$ mapping $[\beta_1, \beta_2, ..., \beta_{i-1}]$ to $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$;
  if $i = l$ then
    if g satisfies P then
      $K := <K, g>$;
      return to search at level s;  (*since $\gamma_s$ is no longer first in its K-orbit*)
    end if;
  else
    for each point $\gamma_i$ in $\overline{\Omega}_P( [\gamma_1, \gamma_2, ..., \gamma_{i-1}] )$ do
      generate( G, P, s, l, i+1, $[\gamma_1, \gamma_2, ..., \gamma_i]$, K );
    end for;
  end if;
end;
```

```
begin
  choose an appropriate base for the problem;
  let $G^{(l)}$ be the largest stabiliser whose elements have property $P$;
  search( $G, P, 1, l, K$ );
end;
```

More on Cosets

This section continues the discussion on using cosets to prune the search. This problem-independent heuristic has already lead to the powerful first in the orbit test. There is still more to be gained, however, even though these improvements are not as significant as the first in the orbit test.

The equivalence of the first element g in the left coset of K and the first base image in $B^{g \times K}$ easily leads to

Proposition 1

An element g with base image $[\gamma_1, \gamma_2, ..., \gamma_k]$ is first in the left coset $g \times K$ if and only if, for each i, γ_i is the first point in the orbit of γ_i under $K_{\gamma_1, \gamma_2, ..., \gamma_{i-1}}$.

Corollary

If γ_i is not the first point in its $K_{\gamma_1, \gamma_2, ..., \gamma_{i-1}}$-orbit then no element g whose base image begins $[\gamma_1, \gamma_2, ..., \gamma_i]$ is first in the left coset $g \times K$.

Corollary

If $K \leq G^{(s)}$ and $g \in G^{(s)}$, and g maps β_s to a point γ_s which is not first in its K-orbit, then g is not first in the left coset $g \times K$.

The base change algorithm does allow us to compute $K_{\gamma_1, \gamma_2, ..., \gamma_{i-1}}$. However, the cost is not trivial. For this reason, almost all backtracks do not use the complete test for an element being first in its left coset of K. One exception is the algorithm for computing the automorphism group of a group. The conclusions of the first corollary also hold when we know some subgroup K' of the stabiliser $K_{\gamma_1, \gamma_2, ..., \gamma_{i-1}}$. A typical example is the subgroup generated by those generators of K which fix $\gamma_1, \gamma_2, ..., \gamma_{i-1}$.

The equivalence of the first element in the right coset and the first image in $B^{K \times g}$ easily leads to

Proposition 2

An element g with base image $[\gamma_1, \gamma_2, ..., \gamma_k]$ is the first element in the right coset $K \times g$ if and only if, for each i, and for each $\gamma \in \beta_i^{K^{(i)}}$, $\gamma_i \leq \gamma^g$.

Corollary

If $\beta_i \in \beta_j^{K^{(j)}}$, for $j < i$, and $\gamma_i < \gamma_j$ then any element g whose base image begins with $[\gamma_1, \gamma_2, ..., \gamma_i]$ is not the first element in the right coset $K \times g$.

In this instance, we know $K^{(i)} = K_{\beta_1, \beta_2, ..., \beta_{i-1}}$ because the backtrack produces a strong generating set of K relative to $B = [\beta_1, \beta_2, ..., \beta_k]$. However, the test requires that later base points β_i be in the basic orbit $\beta_j^{K^{(j)}}$ of K. It is unclear how frequently this occurs.

The only changes are to the procedure *generate*. They are given in Algorithm 5.

Algorithm 5 : Backtrack Search Using Left and Right Cosets

```
Input : a group $G$ with a base $B=[\beta_1, \beta_2, ..., \beta_k]$ and a strong generating set $S$;
    a decidable property $P$ whose elements form a subgroup;

Output : a strong generating set relative to $B$ for the subgroup $K = H(P)$;

procedure generate( $G$ : group; $P$ : property; $s$ : $1..k+1$; $l$ : $1..k+1$; $i$ : $1..k+1$;
                    $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$ : initial segment of base image;
                    var $K$ : group );
(* Generate the coset representatives of $G^{(l)}$ in $G^{(s)}$ whose base image start
  with $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$ and that may have property $P$.
  If one is found then extend $K$, the subgroup of $G^{(s)}$ of elements
  with property $P$ that have already been found, and return to search. *)
begin
  find an element $g \in G$ mapping $[\beta_1, \beta_2, ..., \beta_{i-1}]$ to $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$;
  if $i = l$ then
    if $g$ satisfies $P$ then
      $K := \langle K, g \rangle$;
      return to search at level $s$; (*since $\gamma_s$ is no longer first in its $K$-orbit*)
    end if;
  else
    for each point $\gamma_i$ in $\overline{\Omega}_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}])$ do
      if ($\gamma_i$ is first in $\gamma_i^{K_{\gamma_1, \gamma_2, ..., \gamma_{i-1}}}$)
                and
           ($\gamma_i \geq \gamma_j$, for all $j < i$ where $\beta_i \in \beta_j^{K^{(j)}}$) then
        generate( $G, P, s, l, i+1, [\gamma_1, \gamma_2, ..., \gamma_i], K$ );
      end if;
    end for;
  end if;
end;
```

Preprocessing

There may be many calculations of the approximations $\overline{\Omega}_P$ performed during the backtrack search. Typically, these computations will use the same information. To speed up the computation, this information should be readily available rather than recalculated each time. Also, as we have mentioned earlier, such information is used in the selection of an appropriate base, so its computation at the outset is necessary in any case.

An example, which occurs in the construction of centralizers, are the points in a z-cycle of the same length of the z-cycle of β_i, when $\beta_i \neq \beta_j^g$, for some $j < i$. This set is constant throughout the computation, so it may be calculated at the outset and stored for later repeated use in calculating $\overline{\Omega}_P([\gamma_1, \gamma_2, ..., \gamma_{i-1}])$.

Case Study 1 : Centralizer

The centralizer has been used as our running example, so this section will just be a summary. The property is

$$P_z \ : \ g \in G \ \text{ centralizes } \ z \in G$$

The base is chosen to match the cycles of z, with the longest cycles being matched first. The restrictions on the base images are

$$\overline{\Omega}_{P_z}(\,[\gamma_1, \gamma_2, \ldots, \gamma_{i-1}]\,) \ = \ \{\ \gamma_{i-1}{}^{z}\ \} \cap \left[\Delta^{(i)}\right]^{g},$$

if $\beta_i = \beta_{i-1}{}^{z}$, and

$$\overline{\Omega}_{P_z}(\,[\gamma_1, \gamma_2, \ldots, \gamma_{i-1}]\,) \ = \ \{\ \gamma \ \mid \ \text{the } z\text{-cycles of } \gamma \text{ and } \beta_i \text{ have the same length}\ \}$$

$$\cap \left[\Delta^{(i)}\right]^{g}$$

otherwise.

The known subgroup is $<z>$.

Shortened base images are used if the process of matching the cycles of z eventually considers cycles of length one. In this case, the end of the base will consist of fixed points of z. The stabiliser of the fixed points centralizes z, so we only generate the images of the initial segment of the base.

Preprocessing constructs, for the relevant values of i, the set of points in z-cycles of length $\mid \beta_i{}^{<z>} \mid$,

Case Study 2 : Normalizer

This section considers the normalizer of a subgroup H of G. We assume we have a base and strong generating set for H. The property is

$$P_H \ : \ g \in G \ \text{ normalizes } \ H \leq G$$

The two facts we know about normalizing elements which lead to our choice of a base and the restrictions on the base images are

Lemma 1

If g normalizes a subgroup H then g permutes the orbits of H.

Lemma 2

If g normalizes a subgroup H and g fixes β then g normalizes the stabiliser H_β.

The lemmas imply that, when searching $G^{(s)}$, the elements of the normalizer will normalize each of $H^{(s)}$, $H^{(s-1)}$, ..., $H^{(1)}$ and permute the orbits of each of the subgroups. This means that

(i) the length of the orbit of γ_i is the same as the length of the orbit of β_i, and

(ii) β_i and β_j are in the same orbit if and only if γ_i and γ_j are in the same orbit,

for each subgroup $H^{(s)}, H^{(s-1)}, ..., H^{(1)}$. So we define

$$\begin{aligned}\overline{\Omega}_{P_H}([\gamma_1, \gamma_2, ..., \gamma_{i-1}]) = & \left[\Delta^{(i)}\right]^g \\ & \cap \bigcap_{t=1}^{s} \{ \gamma \in \Omega \mid |\gamma^{H^{(t)}}| = |\beta_i^{H^{(t)}}| \} \\ & \cap \bigcap_{j=1}^{i-1} \{ \gamma_j^{H^{(t)}} \mid 1 \le t \le s \text{ and } \beta_i \in \beta_j^{H^{(t)}} \} \\ & - \bigcup_{j=1}^{i-1} \{ \gamma_j^{H^{(t)}} \mid 1 \le t \le s \text{ and } \beta_i \notin \beta_j^{H^{(t)}} \}\end{aligned}$$

where, as usual, g is an element of G mapping $[\beta_1, \beta_2, ..., \beta_{i-1}]$ to $[\gamma_1, \gamma_2, ..., \gamma_{i-1}]$. In support of these restrictions on the base images, the base is chosen with three objectives in mind :

(1) maximise the number of occurrences of $\beta_i \in \beta_{i-1}^{H^{(i-1)}}$,

(2) minimise the number of $H^{(i)}$-orbits of length $|\beta_i^{H^{(i)}}|$, and

(3) minimise the number of points in the $H^{(i)}$-orbits of length $|\beta_i^{H^{(i)}}|$.

The known subgroup is H.

If the process of choosing a base eventually includes fixed points of H at the end of the base, then only images of an initial segment of the base are generated, since the stabiliser of these fixed points normalizes H. Indeed, the stabiliser *centralizes H.*

The preprocessing stores information about the orbits of $H, H^{(2)}, H^{(3)}, ...$ such as

(a) the length, and a representative point of each orbit,

(b) each orbit as a linked list of points,

(c) the fusion of the orbits of $H^{(t)}$ in $H^{(t-1)}$, for $t > 1$,

(d) which orbits contain the base points, and their lengths, and

(e) which pairs of base points are contained in the same orbit.

Case Study 3 : Intersection

This section considers the intersection of two groups $H1$ and $H2$ of permutations on Ω. We assume we have a base and strong generating set for both groups. The property is

$$P_{H1 \cap H2} \; : \; g \in H1 \text{ and } g \in H2.$$

For either group $H1$ or $H2$ we could generate all the base images, and therefore all the elements, and test if the element was in the other group. However, if we arrange for both groups to have the same base, we can in effect compare base images directly. A base image will then correspond to an element in the intersection only if it can be generated in both $H1$ and $H2$. The search generates both groups simultaneously, or at least enough of both groups to determine the intersection. The restrictions we place on the base images are

$$\overline{\Omega}_{P_{H1 \cap H2}}([\gamma_1, \gamma_2, \ldots, \gamma_{i-1}]) = \left[\Delta 1^{(i)} \right]^{g1} \cap \left[\Delta 2^{(i)} \right]^{g2}$$

where $g1$ (respectively $g2$) is an element of $H1$ (respectively $H2$) mapping the first $i-1$ base points to $[\gamma_1, \gamma_2, \ldots, \gamma_{i-1}]$. (The numbers 1 and 2 also distinguish the basic orbits of the two groups.)

Once the complete base image is generated, both $g1$ and $g2$ are uniquely determined. However, we still must see if they are the same permutation before concluding that the base image gives an element in the intersection.

Case Study 4 : Set Stabiliser

This section considers the stabiliser of a set $\Delta = \{\delta_1, \delta_2, \ldots, \delta_{l-1}\}$ of points. The property is

$$P_\Delta \; : \; g \in G \text{ and } \delta^g \in \Delta, \text{ for all } \delta \in \Delta.$$

The property places restrictions on the images of the points in the set, so the base is chosen to begin with $\delta_1, \delta_2, \ldots, \delta_{l-1}$ and the approximation is

$$\overline{\Omega}_{P_\Delta}([\gamma_1, \gamma_2, \ldots, \gamma_{i-1}]) = \Delta \cap \left[\Delta^{(i)} \right]^{g}.$$

The stabiliser $G^{(l)}$ fixes each point of the set Δ, so it is in the set stabiliser. Therefore, only images of the initial segment of the base are generated.

Summary

The problem-independent heuristics for improving the backtrack search rely on the use of cosets. They are

(1) first in the K-orbit test in *search*,

(2) first in the $K_{\gamma_1, \gamma_2, \ldots, \gamma_{i-1}}$-orbit test in *generate*, and

(3) weak test for first in the right coset of K.

Of these, the first is of great significance, leading to great improvements in the backtrack search.

The problem-dependent heuristics rely on the property P for their details, though their general strategies are applicable to a wide class of problems. They are

(1) restricting the choice of images using $\overline{\Omega}_P$,

(2) choosing a base appropriate to the problem,

(3) using any known subgroups, and

(4) generating images of an initial base segment.

There is a close connection between the first two. Indeed, the effectiveness of (1) depends on (2), and the selection of a base depends on how it will be used in (1). Both offer significant improvements in the backtrack searches that have been discussed. The importance of (3) and (4) depends on the problem.

The algorithms for the construction of centralizers, normalizers, intersections, and set stabilisers have been presented. The techniques of this chapter are also used in the construction of the automorphism group of a group.

Exercises

(1/Moderate) This series of exercises deals with backtrack searches within the symmetries of the projective plane of order two. They have been chosen so that [1,2,4] is an appropriate base, though maybe not the most appropriate. Therefore, no base changes are necessary. In each case, draw the search tree. Choose an appropriate algorithm that uses at least the first in the K-orbit test, the restriction on images, and any known subgroup.

(i) Let $z = (1,2)(4,7)$. Determine the centralizer of z.

(ii) Let $H = <(1,2)(4,7), (4,7)(5,6)>$ of order 4. Determine its normalizer.

(iii) Let $H1 = <(1,2,4,7)(3,6), (2,4,7)(3,5,6), (4,7)(5,6)>$ and let $H2 = <(1,5,7,3)(2,4), (2,3)(4,5), (2,5)(3,4)>$ both of order 24. Determine their intersection.

(iv) Show that the normalizer of $H1$ is $H1$, and the normalizer of $H2$ is $H2$.

(v) Determine the stabiliser of the set $\{1,2,4\}$.

Bibliographical Remarks

Backtrack searches developed as part of the folklore of combinatorial computing. The first reference to the term *backtrack* in this context appears to be R. J. Walker, "*An enumerative technique for a class of combinatorial problems*", **Combinatorial Analysis**, R. E. Bellman and M. Hall, Jr (editors), Proceedings of the Symposium on Applied Mathematics, **10**, 1960, 91-94.

The use of backtrack searches in relation to permutation groups began with the development of the centralizer algorithm in C. C. Sims, "*Determining the conjugacy classes of a permutation group*", **Computers in Algebra and Number Theory** (Proceedings of the Symposium on Applied Mathematics, New York, 1970), G. Birkhoff and M. Hall, Jr (editors), SIAM-AMS Proceedings, volume 4, American Mathematics Society, Providence, Rhode Island, 1971, 191-195. In this paper, Sims places restrictions on the images of the base, chooses an appropriate base, uses the first in the *K*-orbit test, and uses the weak test for first in the right coset. The classification of the first element in the right coset in terms of its base image is also due to this paper. The next paper developed the intersection algorithm : C. C. Sims, "*Computation with permutation groups*", (Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, 1971), S. R. Petrick (editor), Association of Computing Machinery, New York, 1971, 23-28, and the ideas behind the set stabiliser algorithm are found in the lectures Sims gave in Oxford in January and February 1973.

The classification of the first element of the left coset in terms of its base image can be found in G. Butler, "*Computing in permutation and matrix groups II : backtrack algorithm*", Mathematics of Computation **39**, 160 (1982) 671-670. This chapter follows the presentation of this paper, but recasts the algorithms in a modern language and uses recursion. Although this chapter has concentrated on constructing subgroups, the backtrack search can also be used to find single elements with a given property, as discussed in the above paper.

The normalizer algorithm is due to G. Butler, "*Computing normalizers in permutation groups*", Journal of Algorithms **4** (1983) 163-175, and an algorithm for computing the automorphism group of a group using a backtrack search is described in H. Robertz, **Eine Methode zur Berechnung der Automorphismengruppe einer endliche Gruppe**, Diplomarbeit, Aachen, 1976.

Little is known about the analysis of backtrack algorithms. M. Fontet, "*Calcul de centralisateur d'un groupe de permutations*", Bulletin de la Société Mathématique de France Mémoire **49-50** (1977) 53-63, shows that computing the centralizer of an element in the symmetric group is polynomial-time. The intersection algorithm is known to be exponential by C. M. Hoffman, "*On the complexity of intersecting permutation groups and its relationship with graph isomorphism*", Technical Report 4/80, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, 1980. Hoffman presents polynomial algorithms for special cases of the intersection problem. Estimating the running time of a particular backtrack algorithm on a given problem is discussed in D. E. Knuth, "*Estimating the efficiency of backtrack programs*", Mathematics of Computation **29** (1975) 121-136.

A backtrack search, similar to the one described here, is also used to compute automorphism groups of combinatorial objects such as graphs, codes, and Hadamard matrices. Some references are B. D. McKay, "*Computing automorphisms and canonical labelling of graphs*", **Combinatorial Mathematics**, (Proceedings of the International Conference on Combinatorial Theory, Canberra, August 16-27, 1977), D. A. Holton and J. Seberry (editors), Lecture Notes in Mathematics, **686**, Springer-Verlag, Berlin, 1978, 223-232; J. S. Leon, "*An algorithm for computing the automorphism group of a Hadamard matrix*", Journal of Combinatorial Theory, Series A, **27** (1979) 289-306; J. S. Leon, "*Computing automorphism groups of error-correcting codes*", IEEE Transactions on Information Theory, **IT-28** (1982) 496-511; and J. S. Leon, "*Computing automorphism groups of combinatorial objects*", **Computational Group Theory**, (Proceedings of LMS Symposium on Computational Group Theory, Durham, 1982), M. Atkinson (editor), Academic Press, New York, 1984, 321-335. The last reference is a general overview of backtrack techniques for constructing automorphism groups of combinatorial objects.