Journal of Algebra 521 (2019) 481–506

Contents lists available at ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

Minimal and canonical images

Christopher Jefferson [a], Eliza Jonauskyte [a], Markus Pfeiffer [a], Rebecca Waldecker [b,*]

[a] *University of St Andrews, School of Computer Science, North Haugh, St Andrews, KY16 9SX, Scotland, United Kingdom of Great Britain and Northern Ireland*
[b] *Martin-Luther-Universität Halle-Wittenberg, Institut für Mathematik, 06099 Halle, Germany*

A R T I C L E   I N F O

Article history:
Received 1 March 2017
Available online 22 November 2018
Communicated by Gerhard Hiss

Keywords:
Minimal images
Canonical images
Computation
Group theory
Permutation groups

A B S T R A C T

We describe a family of new algorithms for finding the canonical image of a set of points under the action of a permutation group. This family of algorithms makes use of the orbit structure of the group, and a chain of subgroups of the group, to efficiently reduce the amount of search that must be performed to find a canonical image.

We present a formal proof of correctness of our algorithms and describe experiments on different permutation groups that compare our algorithms with the previous state of the art.

* Corresponding author.
E-mail addresses: caj21@st-andrews.ac.uk (C. Jefferson), ej31@st-andrews.ac.uk (E. Jonauskyte), markus.pfeiffer@st-andrews.ac.uk (M. Pfeiffer), rebecca.waldecker@mathematik.uni-halle.de (R. Waldecker).
URLs: http://caj.host.cs.st-andrews.ac.uk/ (C. Jefferson), https://www.morphism.de/~markusp/ (M. Pfeiffer), http://conway1.mathematik.uni-halle.de/~waldecker/index-english.html (R. Waldecker).

https://doi.org/10.1016/j.jalgebra.2018.11.009

1. Background

Many combinatorial and group theoretical problems [14,4,2] are equivalent to finding, given a group G that acts on a finite set Ω and a subset $X \subseteq \Omega$, a partition of X into subsets that are in the same orbit of G.

We can solve such problems by taking two elements of X and searching for an element of G that maps one to the other. However, this requires a possible $O(|X|^2)$ checks, if all elements of X are in different orbits.

Given a group G acting on a set Ω, a canonical labelling function maps each element of Ω to a distinguished element of its orbit under G. Using a canonical labelling function we can check if two members of Ω are in the same orbit by applying the canonical labelling function to both and checking if the results are equal. More importantly, we can solve the problem of partitioning X into orbit-equivalent subsets by performing $O(|X|)$ canonical image calculations. Once we have the canonical image of each element, we can organize the canonical images into equivalence classes by sorting in $O(|X|log(|X|))$ comparisons, or expected $O(|X|)$ time by placing them into a hash table. This is because checking if two elements are in the same equivalence class is equivalent to checking if their canonical images are equal.

The canonical image problem has a long history. Jeffrey Leon [9] discusses three types of problems on permutation groups – subgroup-type problems (finding the intersection of several groups), coset-type problems (deciding whether or not the intersection of a series of cosets is empty, and if not, finding their intersection) and canonical-representative-type problems. He claims to have an algorithm to efficiently solve the canonical-representative problem, but does not discuss it further. His comments have inspired mathematicians and computer scientists to work on questions related to minimal images and canonical images.

One of the most well-studied canonical-image problems is the canonical graph problem. Current practical systems derive from partition refinement techniques, which were first practically used for graph automorphisms by McKay [11] in the `Nauty` system. There have been a series of improvements to this technique, including `Saucy` [1], `Bliss` [8] and `Traces` [12]. A comparison of these systems can be found in [12].

We cannot, however, directly apply the existing work for graph isomorphism to finding canonical images in arbitrary groups. The reason is that McKay's Graph Isomorphism algorithm only considers finding the canonical image of a graph under the action of the full symmetric group on the set of vertices. Many applications require finding canonical images under the action of subgroups of the full symmetric group.

One example of a canonical labelling function is, given a total ordering on X, to map each value of X to the smallest element in its orbit under G. This *Minimal image problem* has been treated by Linton in [10]. Pech and Reichard [13] apply techniques similar to Linton's to enumerate orbit representatives of subsets of Ω under the action of a permutation group on Ω. Linton gives a practical algorithm for finding the smallest image of a set under the action of a given permutation group. Our new algorithm, inspired by Linton's work, is designed to find canonical images: we extend and generalize Linton's technique using a new orbit-based counting technique. In this paper we first introduce some notation and explain the concepts that go into the algorithm, then we prove the necessary results and finish with experiments that demonstrate how this new algorithm is superior to the previously published techniques.

2. Minimal and canonical images

Throughout this paper, Ω will be a finite set, G a subgroup of $Sym(\Omega)$, and Ω will be ordered by some (not necessarily total) order $\leqslant$. If $\alpha \in \Omega$, then we denote the orbit of α under G by α^G. Similarly, if $A \subseteq \Omega$ and $g \in G$, then $A^g := \{a^g \mid a \in A\}$ and $A^G := \{A^g \mid g \in G\}$.

In this paper, we want to efficiently solve the problem of deciding, given two subsets $A, B \subseteq \Omega$, if $A \in B^G$. We do this by defining a canonical image:

Definition 2.1. A **canonical labelling function** C for the action of G on a set Ω is a function $C : \mathcal{P}(\Omega) \to \mathcal{P}(\Omega)$ such that, for all $A \subseteq \Omega$, it is true that

- $C(A) \in A^G$, and
- $C(A^g) = C(A)$ for all $g \in G$.

In this situation we call $C(A)$ the **canonical image** of $A \subseteq \Omega$ (with respect to G in this particular action).

Further, we say that $g_A \in G$ is a **canonizing element** for A if and only if $A^{g_A} = C(A)$.

A canonical image can be seen as a well-defined representative of a G-orbit on Ω with respect to the defined action. While in this paper we will only consider the action of G on a set of subsets of Ω, canonical images are defined similarly for any group and action. In practice we want to be able to find canonical images effectively and efficiently. In some situations we are interested in computing the canonizing element, which might not be uniquely determined. Our algorithms will always produce a canonizing element as a byproduct of search. We choose to make this explicit here to make the exposition clearer.

Minimal images are a special type of canonical image.

Remark 2.2. Suppose that $\preccurlyeq$ is a partial order on Ω such that any two elements in the same orbit can be compared by $\preccurlyeq$.

Let $\mathrm{Min}_{\preccurlyeq}$ denote the function that, for all $\omega \in \Omega$, maps ω to the smallest element in its orbit. Then $\mathrm{Min}_{\preccurlyeq}$ is a canonical labelling function.

In practical applications we are interested in more structure, namely in structures that G can act on naturally via the action on a given set Ω. These structures include subsets of Ω, graphs with vertex set Ω, sets of maps with domain or range Ω, and so on.

In this paper, our main application will be finding canonical images when acting on a set of subsets of Ω.

Definition 2.3. Suppose that $\leqslant$ is a total order of Ω. Then we introduce a total order $\preccurlyeq$ on $\mathcal{P}(\Omega)$ as follows:

We say that A **is less than** B and write $A \preccurlyeq B$ if and only if A contains an element a such that $a \notin B$ and $a \leqslant b$ for all $b \in B \setminus A$.

Example 2.4. Let $\Omega := \{1, 2, 3, 4, 5, 6, 7\}$ with the natural order and let $A := \{1, 3, 4\}$, $B := \{3, 5, 7\}$, $C := \{3, 6, 7\}$, $D := \{1, 3\}$ and $E := \{2\}$.

Now $A \preccurlyeq B$, because $1 \in A$, $1 \notin B$ and 1 is smaller than all the elements in B, in particular those not in A. Moreover $A \preccurlyeq C$ for the same reason. Furthermore, $B \preccurlyeq C$, because $5 \in B$, $5 \notin C$, and if we look at $C \setminus B$, then this only contains the element 6 and 5 is smaller.

Next we consider A and D. As $4 \in A \setminus D$ and $D \setminus A = \varnothing$, we see that $A \preccurlyeq D$. Also $A \preccurlyeq E$ because $1 \in A \setminus E$ and 1 is smaller than all elements in $E \setminus A = E$. Finally $E \preccurlyeq B$ because $2 \in E \setminus B = E$ and 2 is smaller than all elements in $B \setminus E = B$.

Remark 2.5. The example illustrates that this new order introduced above reduces to lexicographical order for sets of the same size. But for sets of different sizes, it might seem counter-intuitive. Our reason for choosing this different ordering is that it satisfies the following property:

If $n \in \mathbb{N}$ and A and B are sets of positive integers, then $A \cap \{1, \dots, n\} < B \cap \{1, \dots, n\}$ implies $A < B$. This means that, when building A and B incrementally, we know if A is smaller than B as soon as we find the first integer that is contained in one of the sets but not in the other. This is not true for lexicographic ordering of sets, as $\{1\} < \{1, 2\}$ but $\{1, 1000\} > \{1, 2, 1000\}$.

If G is a subgroup of $Sym(\Omega)$ and $\omega \in \Omega$, then we denote by G_ω the point stabilizer of ω in G. For distinct elements $x, y \in \Omega$, we denote by $G_{x \mapsto y}$ the set of all elements of G that map x to y. This set may be empty.

We remark that the above information is readily available from a stabilizer chain for the group G, which can be calculated efficiently. For further details we refer the reader to [5]. We now introduce some notation and then prove a basic result about cosets.

Definition 2.6. Let G be a permutation group acting on a totally ordered set $(\Omega, \leqslant)$, and let $\preccurlyeq$ denote the induced ordering as explained in Definition 2.3. Let H be a subset of G and $S \subseteq \Omega$. Then we define **the minimal image of** S **under** H to be the smallest element in the set $\{S^h \mid h \in H\}$ with respect to $\preccurlyeq$.

In order to simplify notation, we will from now on write $\leqslant$ for the induced order and then we write $\mathrm{Min}(H, S, \leqslant)$ for the minimal image of S under H. We point out for clarity that H will usually be a subgroup or a coset.

Lemma 2.7. *Let G be a permutation group acting on a totally ordered set $(\Omega, \leqslant)$, and let H be a subgroup of G and $S \subseteq \Omega$. Then the following hold for all $x, y \in \Omega$:*

(i) *For all $\sigma \in H_{x \mapsto y}$ it is true that $\sigma \cdot H_y = H_{x \mapsto y} = H_x \cdot \sigma$.*
(ii) *If $\sigma \in H_{x \mapsto y}$, then $\mathrm{Min}(\sigma \cdot H, S, \leqslant) = \mathrm{Min}(H, S^\sigma, \leqslant)$.*

Proof. If $\sigma \in H_{x \mapsto y}$, then multiplication by σ from the right or left is a bijection on H, respectively. For all $\alpha \in H_x$ we have that $\alpha \cdot \sigma$ maps x to y and for all $\beta \in H_y$ we see that $\sigma \cdot \beta$ also maps x to y. This implies the first statement.

For (ii) we just look at the definition: $\mathrm{Min}(\sigma \cdot H, S, \leqslant)$ denotes the smallest element in the set $\{S^{\sigma \cdot h} \mid h \in H\}$ and $\mathrm{Min}(H, S^\sigma, \leqslant)$ denotes the smallest element in the set $\{(S^\sigma)^h \mid h \in H\}$, which is the same set. □

2.1. Worked example

We will find minimal, and later canonical, images using similar techniques to Linton in [10]. This algorithm splits the problem into small sub-problems, by splitting a group into the cosets of a point stabilizer. We will begin by demonstrating this general technique with a worked example.

Example 2.8. In the following example we will look at $\Omega = \{1, 2, 3, 4, 5, 6\}$, the subgroup $G = \langle (14)(23)(56), (126) \rangle \leqslant S_6$, and $S = \{2, 3, 5\}$. We intend to find the minimal image $\mathrm{Min}(G, S, \leqslant)$, where the ordering on subsets of Ω is the induced ordering from $\leqslant$ on Ω as explained in Definition 2.3.

We split our problem into pieces by looking at cosets of $G_1 = \langle (3, 4, 5) \rangle$. The minimal image of S under G will be realized by an element contained in (at least) one of the cosets of G_1, so if we find the minimal image of S under elements in each coset, and then take the minimum of these, we will find the global minimum.

Lemma 2.7 gives that, for all $g \in G$, it holds that $\mathrm{Min}(g \cdot G_1, S, \leqslant) = \mathrm{Min}(G_1, S^g, \leqslant)$, and so we can change our problem from looking for the minimal image of S with respect to cosets of G_1 to looking at images of S^g under elements of G_1 where g runs over a set of coset representatives of G_1 in G.

For each $i \in \{1, \ldots, 6\}$ we need an element $g_i \in G_{i \mapsto 1}$ (where any exist), so that we can then consider S^{g_i}.

We choose the elements id, (162), (146523), (14)(23)(56), (142365) and (126) and obtain six images of S:

$$\{2, 3, 5\}, \{1, 3, 5\}, \{3, 1, 2\}, \{3, 2, 6\}, \{3, 6, 1\}, \{6, 3, 5\}.$$

As we are looking at the images of these sets under G_1, we know that all images of a set containing 1 will contain 1, and all images of a set not containing 1 will not contain 1. From Definition 2.3, all subsets of $\{1, \ldots, 6\}$ containing 1 are smaller than all subsets not containing 1. This means that we can filter our list down to $\{1, 3, 5\}, \{3, 1, 2\}$ and $\{3, 6, 1\}$.

Furthermore, G_1 fixes 2, so by the same argument we can filter our list of sets not containing 2, leaving only $\{3, 1, 2\}$. The minimal image of this under G_1 is clearly $\{3, 1, 2\}$ (in this particular case we could of course also have stopped as soon as we saw $\{3, 1, 2\}$, as this is the smallest possible set of size 3).

Now, let us consider what would happen if the ordering of the integers was reversed, so we are looking for $\mathrm{Min}(G, S, \geqslant)$, again with the induced ordering.

For the same reasons as above, we begin by calculating $G_6 = \langle(3, 5, 4)\rangle$ and by finding images of S for some element from each coset of G_6 in G.

An example of six images is

$$\{1, 5, 4\}, \{6, 5, 3\}, \{4, 6, 1\}, \{5, 1, 2\}, \{3, 2, 6\}, \{2, 3, 5\}.$$

We can ignore anything that does not contain 6, so we are left with:

$$\{6, 5, 3\}, \{4, 6, 1\}, \{3, 2, 6\}.$$

As 5 is not fixed by G_6, we can not reason about the presence or absence of 5 in our sets. There is an image of every set that contains 5, and there are even two distinct images of $\{6, 5, 3\}$ that contain 5. Therefore we must continue our search by considering $G_{6,5}$.

Application of an element from each coset of $G_{6,5}$ to S generates nine sets, of which four contain the element 5. In fact we reach $\{6, 3, 5\}, \{6, 5, 4\}$ from the set $\{6, 4, 3\}$, we reach $\{5, 6, 1\}$ from the set $\{4, 6, 1\}$ and we reach $\{5, 2, 6\}$ from the set $\{3, 2, 6\}$. From these we extract the minimal image $\{6, 5, 4\}$.

In this example, different orderings of $\{1, 2, 3, 4, 5, 6\}$ produced different sized searches, with different numbers of levels of search required.

3. Minimal images under alternative orderings of Ω

As was demonstrated in Example 2.8, the choice of ordering of the set our group acts on influences the size of the search for a minimal image. In this section we will show how to create orderings of Ω that, on average, reduce the size of search for a minimal image.

We begin by showing how large a difference different orderings can make. We do this by proving that, for any choice $\preccurlyeq$ of ordering of Ω, group G and any input set S, we can construct a minimal image problem that is as hard as finding $\mathrm{Min}(G, S, \preccurlyeq)$, but where reversing the ordering on Ω makes the problem trivial.

We make this more precise: Given $n \in \mathbb{N}$, a permutation group G on $\{1, \dots, n\}$ with some ordering $\leqslant$ and a subset $S \subseteq \{1, \dots, n\}$, we construct a group H and a set T such that

$$\mathrm{Min}(G, S, \leqslant) = \mathrm{Min}(H, T, \leqslant) \cap \{1, \dots, n\},$$

showing that finding $\mathrm{Min}(H, T, \leqslant)$ is at least as hard as finding $\mathrm{Min}(G, S, \leqslant)$. On the other hand, we will show that $\mathrm{Min}(H, T, \geqslant) = T$ and that this can be deduced without search. This is done in Lemma 3.5. An example along the way will illustrate the construction.

Definition 3.1. We fix $n \in \mathbb{N}$ and we let $k \in \mathbb{N}$. For all $j \in \mathbb{N}$ we define $q(j) \in \mathbb{N}$ (where q stands for "quotient") and $r(j) \in \{1, \ldots, n\}$ (where r stands for "remainder") such that $j = q(j) \cdot n + r(j)$.

Let $\mathrm{ext} : G \to S_{k \cdot n}$ be the following map: For all $g \in G$ and all $j \in \{1, \ldots, k \cdot n\}$, the element $\mathrm{ext}(g)$ maps j to $q(j) \cdot n + r(j)^g$.

Example 3.2. Let $n = 4$ and $G = S_4$. Then we extend the action of G to the set $\{1, \ldots, 12\}$ using the map ext.

For example $g = (134)$ maps 4 to 1. We write $12 = 2 \cdot 4 + 4$ and then it follows that $\mathrm{ext}(g)$ maps 12 to $2 \cdot 4 + 4^g = 8 + 1 = 9$. In fact g acts simultaneously on the three tuples $(1, 2, 3, 4)$, $(5, 6, 7, 8)$ and $(9, 10, 11, 12)$ as it does on $(1, 2, 3, 4)$.

Definition 3.3. Fixing $n, k \in \mathbb{N}$ and a subgroup G of S_n, and using the map ext defined above, we say that H **is the extension of** G **on** $\{1, \ldots, k \cdot n\}$ if and only if $H = \{\mathrm{ext}(g) \mid g \in G\}$ is the image of G under the map ext.

The extension H of G on a set $\{1, \ldots, k \cdot n\}$ is a subset of $S_{k \cdot n}$. We show now that even more is true:

Lemma 3.4. *Let $n, k \in \mathbb{N}$ and $G \leqslant S_n$. Then the extension of G onto $\{1, \ldots, k \cdot n\}$ is a subgroup of $S_{k \cdot n}$ that is isomorphic to G.*

Proof. Let $H := \mathrm{ext}(G)$ be the image of G under the map ext and let $a, b \in G$ be distinct. Then let $j \in \{1, \ldots, n\}$ be such that $j^a \neq j^b$. By definition $\mathrm{ext}(a)$ and $\mathrm{ext}(b)$ map j in the same way that a and b do, so we see that $\mathrm{ext}(a) \neq \mathrm{ext}(b)$. Hence the map ext is injective. Therefore $\mathrm{ext} : G \to H$ is bijective.

Next we let $a, b \in G$ be arbitrary and we let $j \in \{1, \ldots, k \cdot n\}$. Then the composition ab is mapped to $\mathrm{ext}(ab)$, which maps j to $q(j) \cdot n + r(j)^{ab}$. Now $r(j)^{ab} = (r(j)^a)^b$ and therefore the composition $\mathrm{ext}(a)\,\mathrm{ext}(b) \in S_{k \cdot n}$ maps j to $(q(j) \cdot n + r(j)^a)^{\mathrm{ext}(b)} = q(j) \cdot n + (r(j)^a)^b$. This is because $r(j)^a \in \{1, \ldots, n\}$.

Hence $\mathrm{ext}(ab) = \mathrm{ext}(a)\,\mathrm{ext}(b)$. That implies ext is a group homomorphism and hence that G and its image are isomorphic. □

Lemma 3.5. *Let $n \in \mathbb{N}$ and $G \leqslant S_n$. Let H denote the extension of G on $\{1, \ldots, (n+1) \cdot n\}$ and let $S \subseteq \{1, \ldots, n\}$. Let $T := S \cup \{l \cdot n + l \mid l \in \{1, \ldots, n\}\}$, let $\leqslant$ denote the natural ordering of the integers, and let $\geqslant$ denote its reverse. For simplicity we use the same symbols for the ordering induced on $\mathcal{P}(\Omega)$, respectively. Then*

- $\mathrm{Min}(H, T, \leqslant) \cap \{1, \ldots, n\} = \mathrm{Min}(G, S, \leqslant)$.
- $\mathrm{Min}(H, T, \geqslant) = T$.

Proof. Let $h \in H$. Then by construction h stabilizes the partition

$$[1, \ldots, n \mid n+1, \ldots, 2n \mid \cdots \mid n \cdot n, \ldots, (n+1) \cdot n].$$

Moreover, for all $i \in \{1, \ldots, n\}$ and $g \in G$ we have that $i^g = i^{\mathrm{ext}(g)}$ and so Lemma 3.4 implies that

$$\begin{aligned}\mathrm{Min}(G, S, \leqslant) &= \min_{\leqslant}\{S^g \mid g \in G\} \cap \{1, \ldots, n\} \\ &= \min_{\leqslant}\{S^{\mathrm{ext}(g)} \mid g \in G\} \cap \{1, \ldots, n\} \\ &= \min_{\leqslant}\{S^h \mid h \in H\} \cap \{1, \ldots, n\} \\ &= \mathrm{Min}(H, T, \leqslant) \cap \{1, \ldots, n\}.\end{aligned}$$

This proves the first statement.

For the second statement we notice that $(n+1) \cdot n$ is now the smallest element of T, and it cannot be mapped to anything smaller, because it also is the smallest element available. So if we let $h \in H$ be such that $T^h = \mathrm{Min}(H, T, \geqslant)$, then h fixes the point $n(n+1)$. By definition of the extension, it follows that h also fixes $k \cdot n$ for all $k \in \{1 \ldots n\}$. The next point of T under the ordering is $n^2 - 1$. It cannot be mapped by h to n^2, because n^2 is already fixed, hence h has to fix $n^2 - 1$, too.

Arguing as above it follows that all points are fixed by h, thus in particular $\mathrm{Min}(H, T, \geqslant) = T$, as stated. Furthermore, any algorithm that stepped through the elements of T in the order we describe would find this smallest element without having to perform a branching search, as at each step there is no choice on which element of T is the next smallest. □

3.1. Comparing minimal images cheaply

We describe some important aspects of Linton's algorithm for computing the minimal image of a subset of Ω.

Definition 3.6. Suppose that $(\Omega, \leqslant)$ is a totally ordered set and that $G \leqslant Sym(\Omega)$. Then $\mathrm{Orb}(G)$ denotes the list of orbits of G on Ω. This list of orbits is ordered with respect to the smallest element in each orbit under $\leqslant$.

A G-orbit will be called a **singleton** if and only if it has size 1.

If $S \subseteq \Omega$, then we say that a G-orbit is **empty in S** if and only if it is disjoint from S as a set, and we say that it is **full in S** if and only if it is completely contained in S.

Example 3.7. Let $\Omega := \{1, \ldots, 8\}$, with the natural ordering on the integers, let $G := \langle(1,4), (2,8), (5,6), (7,8)\rangle$ and let $S := \{1,3,5,6\}$.

Then $\mathrm{Orb}(G) = [\{1,4\}, \{2,7,8\}, \{3\}, \{5,6\}]$ because this list contains all the G-orbits and they are ordered by the smallest element in each orbit, namely 1 in the first, 2 in the second, 3 in the third, which is a singleton, and 5 in the last (because 4 is already in an earlier orbit).

The orbits $\{3\}$ and $\{5,6\}$ are full in S, the orbit $\{2,7,8\}$ is empty in S and $\{1,4\}$ is neither.

Lemma 3.8. *Suppose that $(\Omega, \leqslant)$ is a totally ordered finite set and that $G \leqslant Sym(\Omega)$. If $\mathrm{Min}(G, S, \leqslant) = \mathrm{Min}(G, T, \leqslant)$ and $\omega \in \Omega$, then ω^G is empty in S if and only if it is empty in T, and ω^G is full in S if and only if it is full in T.*

Proof. Let $\omega \in \Omega$ and suppose that ω^G is empty in S. As ω^G is closed under the action of G, and $S_0 := \mathrm{Min}(G, S, \leqslant)$ is an image of S under the action of G, we see that ω^G is empty in S_0 and hence in $T_0 := \mathrm{Min}(G, T, \leqslant)$. Thus ω^G is empty in T, which is an image of T under the action of G. The same arguments work vice versa.

Next we suppose that ω^G is full in S. Then it is full in $S_0 = T_0$ and hence in T, and the same way we see the converse. □

We can now prove Theorem 3.9, which provides the main technique used to reduce search. This allows us to prove that the minimal image of some set S will be smaller than or equal to the minimal image of a set T, without explicitly calculating the minimal image of either S or T.

Theorem 3.9. *Suppose that G is a permutation group on a totally ordered finite set $(\Omega, \leqslant)$ and that S and T are two subsets of Ω where $|S| = |T|$.*

Suppose further that o is the first orbit in the list $\mathrm{Orb}(G)$ that is neither full in both S and T nor empty in both S and T. If o is empty in T, but not in S, then $\mathrm{Min}(G, S, \leqslant)$ is strictly smaller than $\mathrm{Min}(G, T, \leqslant)$.

Proof. Suppose that o is empty in T, but not in S. Then o is empty in $T_0 := \mathrm{Min}(G, T, \leqslant)$, but not in $S_0 := \mathrm{Min}(G, S, \leqslant)$, and in particular T_0 and S_0 are distinct, as we have seen in Lemma 3.8.

Let α denote the minimum of the orbit o with respect to $\leqslant$ and let $\omega \in \Omega$. If $\omega < \alpha$, then $\omega \notin o$, so the orbit ω^G appears in the list $\mathrm{Orb}(G)$ *before* o does. Then the choice of o implies that one of the following two cases holds:

(i) ω^G is full in both S and T. In particular, for all $g \in G$ we have that $\omega \in S^g \cap T^g$.
(ii) ω^G is empty in both S and T. In particular, for all $g \in G$ we have that $\omega^G \cap S^g = \varnothing$ and $\omega^G \cap T^g = \varnothing$.

If S_0 contains an element $\omega \in \Omega$ such that $\omega < \alpha$, then Case (i) above holds and $\omega \in T_0$. So $S_0 \cap \{\omega' \in \Omega \mid \omega' < \alpha\} = T_0 \cap \{\omega' \in \Omega \mid \omega < \alpha\}$.

Since S_0 and T_0 are distinct, they must differ amongst the elements at least as large as α and, since they have the same cardinality, the smallest such element determines which of S_0 and T_0 is smaller.

We recall that $o = \alpha^G$ is empty in T and non-empty in S, so there exists some $g \in G$ such that $\alpha \in S^g$. Then $S^g = S_0$ and $\alpha \notin T_0$, so S_0 is strictly smaller that T_0. □

Here is an example how to use Theorem 3.9.

Example 3.10. Let $\Omega := \{1, \ldots, 10\}$ with natural ordering, and let $G := \langle(12), (45), (56), (89)\rangle$. We consider the sets $S := \{3, 6, 7\}$ and $T := \{3, 7, 9\}$ and we want to calculate the smallest of $\mathrm{Min}(G, S)$ and $\mathrm{Min}(G, T)$. Hence, we want to know which one is smaller as cheaply as possible, to avoid superfluous calculations.

We first list the orbits of G: $[\{1, 2\}, \{3\}, \{4, 5, 6\}, \{7\}, \{8, 9\}, \{10\}]$.

Going through the orbits as listed, we see that the first one is empty in S and T, the second one is full in S and T, and the third one gives a difference for the first time. It is empty in T, but not in S, so Theorem 3.9 yields that the minimal image of S is strictly smaller than that of T.

3.2. Static orderings of Ω

In this section we look at which total ordering of Ω should be used to minimize the amount of time taken to find minimal images of subsets of Ω.

Given a group G we will choose an ordering on Ω such that orbits with few elements appear as early as possible. In particular, singleton orbits should appear first.

This is justified by the fact that singleton orbits are always either full or empty. Also, we would expect smaller orbits to be more likely to be empty or full than larger orbits. This means that small orbits placed early in the ordering of Ω are more likely to lead to Theorem 3.9 being applicable, leading to a reduction in search.

Algorithm 1 heuristically chooses a new ordering for an ordered set Ω, only depending on the group G, under the assumption that the algorithm that computes minimal images will pick a point from a smallest non-singleton orbit to branch on. This will not always be true – in practice Linton's algorithm branches on the first orbit which contains some point contained in one of the current candidates for minimal image.

However, we will show that in Section 5 that Algorithm 1 produces substantially smaller, and therefore faster, searches in practice.

It is not necessary in Line 8 of Algorithm 1 to choose the smallest element of *Points*, choosing an arbitrary element will, on average, perform just as well. By fixing which point is chosen, we ensure that independent implementations will produce the same ordering and therefore the same canonical image.

Algorithm 1 FixedMinOrbit.

1: **procedure** MinOrbitOrder(Ω, G)
2: $\quad$ $Remain := \Omega$
3: $\quad$ $Order := []$
4: $\quad$ $H := G$
5: $\quad$ **while** $|Remain| > 0$ **do**
6: $\quad\quad$ $OrbSize := \text{Min}\left\{|o| \;\middle|\; o \in \text{Orb}(H), o \cap Remain \neq \emptyset\right\}$
7: $\quad\quad$ $Points := \left\{o \;\middle|\; o \in \text{Orb}(H), |o| = OrbSize, o \cap Remain \neq \emptyset\right\}$
8: $\quad\quad$ $MinPoint := \text{Min}\{x \mid o \in Points, x \in o\}$
9: $\quad\quad$ $Remain := Remain \backslash \{MinPoint\}$
10: $\quad\quad$ $Add(Order, MinPoint)$
11: $\quad\quad$ $H := G_{MinPoint}$
12: $\quad$ **return** $Order$

We will also consider one simple modification of Algorithm 1, namely `FixedMaxOrbit` (which is the same as `FixedMinOrbit`) with line 6 changed to pick orbits of maximum size.

If our intuition about Theorem 3.9 is correct, then `MaxOrbit` should almost always produce a larger search than `MinOrbit` or a random ordering of Ω.

3.3. Implementing alternative orderings of Ω

Having calculated an alternative *Order* using **FixedMinOrbit** or **FixedMaxOrbit**, we could create a version of **MinimalImage** which accepted an explicit ordering. However, rather than editing the algorithm, we can instead perform a pre-processing step, using Lemma 3.11.

Lemma 3.11. *Consider a group G that acts on $\Omega = \{1, \dots, n\}$ and a permutation $\sigma \in Sym(\Omega)$. We define an ordering $\leqslant_\sigma$ on $\{1, \dots, n\}$, where for all $x, y \in \Omega$ we have that $x \leqslant_\sigma y$ if and only if $x^\sigma \leqslant y^\sigma$.*

For the induced orderings $\preccurlyeq$ and $\preccurlyeq_\sigma$ on subsets of Ω as in Definition 2.3 it holds that

$$X \preccurlyeq_\sigma Y \Leftrightarrow X^\sigma \preccurlyeq Y^\sigma$$

for all subsets X and Y of Ω, and hence (simplifying notation)

$$\text{Min}(G, S, \preccurlyeq_\sigma) = \text{Min}(G^\sigma, S^\sigma, \preccurlyeq)^{\sigma^{-1}}.$$

Proof. Following Definition 2.3, $X \preccurlyeq_\sigma Y$ if and only if there is an $x \in X$ such that $x \notin Y$ and for all $y \in Y \backslash X$ it holds that $x \leqslant_\sigma y$. By definition of $\leqslant_\sigma$, this is the case whenever $x^\sigma \leqslant y^\sigma$, and since $x^\sigma \in X^\sigma$ and for all y^σ in $Y^\sigma \backslash X^\sigma$ it holds that $x^\sigma \leqslant y^\sigma$, it follows that $X^\sigma \preccurlyeq Y^\sigma$.

Consider the map $\varphi_\sigma : S^G \to (S^\sigma)^{G^\sigma}$ that maps sets $X \in S^G$ to $X^\sigma \in (S^\sigma)^{G^\sigma}$. This map is bijective, and by the above it respects the ordering, so the second claim follows. $\square$

Lemma 3.11 gives an efficient method to calculate minimal images under different orderings without having to alter the underlying algorithm. The most expensive part of this algorithm is calculating G^σ, but this is still very efficiently implemented in systems such as GAP, and also can be cached so it only has to be calculated once for a given G and σ.

4. Dynamic ordering of Ω

In Section 3.2, we looked at methods for choosing an ordering for Ω that allows a minimal image algorithm to search more quickly. There is a major limitation to this technique – it does not make use of the sets whose canonical image we wish to find.

In this section, instead of producing an ordering ahead of time, we will incrementally define the ordering of Ω as the algorithm progresses. At each stage we will consider exactly which extension of our partially constructed ordering will lead to the smallest increase in the number of sets we must consider.

We are not free to choose our ordering arbitrarily as we must still map two sets in the same orbit of G to the same canonical image. However, we can use different orderings for sets that are in different orbits of G.

Firstly, we will explain how we build the orderings that our algorithm uses.

4.1. Orderings

When building canonical images, we build orderings as the algorithm progresses. We represent these partially built orderings as ordered partitions.

Definition 4.1. Let $k \in \mathbb{N}$ and let $P = [X_1, \ldots, X_k]$ be an ordered partition of $\mathcal{P}(\Omega)$. Then, given two subsets S and T of Ω, we write $S <_P T$ if and only if the cell that contains S occurs before the cell that contains T in P.

We say that P is *G-invariant* if and only if for all $i \in \{1, \ldots, k\}$ and $g \in G$ it holds that $S \in X_i$ if and only if $S^g \in X_i$.

A *refinement* of an ordered partition $[X_1, \ldots, X_k]$ is an ordered partition $[Y_{1,1}, Y_{1,2}, \ldots, Y_{k,l}]$ where $l \in \mathbb{N}$ and such that, for all $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, l\}$, we have that $Y_{i,j} \subseteq X_i$.

A *completion* of an ordered partition $X = [X_1, \ldots, X_k]$ is a refinement where every cell is of size one. Given an ordering for Ω, the *standard completion* of an ordered partition X orders the members of each cell of X using the ordering on sets from Definition 2.3.

In our algorithm we need a completion of an ordered partition, but the exact completion is unimportant – it is only important that, given an ordered partition X, we always return the same completion. For this reason we define the standard completion of an ordered partition.

Example 4.2. Let $G := \langle(12)\rangle \leqslant S_3$ and $\Omega := \{1,2,3\}$. Moreover let
$P := [\{\}, \{1\}, \{2\}, \{1,3\}, \{2,3\} \mid \{3\}, \{1,2\}, \{1,2,3\}]$ be an ordered partition of $\mathcal{P}(\Omega)$.

The orbits of G on Ω are $\{1,2\}$ and $\{3\}$. In particular all elements of G stabilize the partition P.

The ordered partition $Q := [\{1,3\}, \{2\} \mid \{\}, \{1\}, \{2,3\} \mid \{3\}, \{1,2\}, \{1,2,3\}]$ is a refinement of P that is not G-invariant.

To see this, we let $g := (12) \in G$. We have that $\{1,3\}$ is in the first cell of Q, but $\{1,3\}^g = \{2,3\}$ is not in the first cell.

In Example 4.2 we only considered a very small group, because the size of $\mathcal{P}(\Omega)$ is $2^{|\Omega|}$. In practice we will not explicitly create ordered partitions of $\mathcal{P}(\Omega)$, but instead store a compact description of them from which we can deduce the cell that any particular set is in.

In this paper, we will consider two methods of building and refining ordered partitions. We first define the orbit count of a set, which we will use when building refiners.

Definition 4.3. Let G be a group acting on an ordered set Ω, and $S \subseteq \Omega$. Define the **orbit count** of S in G, denoted $\mathrm{Orbcount}(G, S)$ as follows: Given the list $\mathrm{Orb}(G)$ of orbits of G on Ω sorted by the their smallest member, the list $\mathrm{Orbcount}(G, S)$ contains the size of the intersection $|o \cap S|$ in place of $o \in \mathrm{Orb}(G)$.

We will see the practical use of Orbcount in Lemma 5.1.

Lemma 4.4. *Suppose that $(\Omega, \leqslant)$ is a totally ordered finite set and that $G \leqslant Sym(\Omega)$. Suppose further that $S, T \subseteq \Omega$ and that there is some $g \in G$ such that $S^g = T$. Then* $\mathrm{Orbcount}(G, S) = \mathrm{Orbcount}(G, T)$.

Proof. Let $o \in \mathrm{Orb}(G)$ and $g \in G$ with $S^g = T$. Then $o^g = o$ and $\alpha \in (o \cap S)$ if and only if $\alpha^g \in (o \cap S)^g$, if and only if $\alpha^g \in (o^g \cap S^g) = (o \cap T)$. □

Definition 4.5. Let P be an ordered partition of $\mathcal{P}(\Omega)$.

- If $\alpha \in \Omega$, then the **point refinement** of P by α is the ordered partition Q defined in the following way: Each cell X_i of P is split into two cells, namely the cell $\{S \mid S \in X_i, \alpha \in S\}$, and the cell $\{S \mid S \in X_i, \alpha \notin S\}$. If one of these sets is empty, then X_i is not split.
- If $G \leqslant Sym(\Omega)$ and $C = \mathrm{Orbcount}(G, T)$ for some set $T \subseteq \Omega$, then the **orbit refinement** of P by C is the ordered partition Q defined as follows: Each cell X_i of P is split into two cells, namely $\{S \mid S \in X_i, \mathrm{Orbcount}(G, S) = C\}$, and $\{S \mid S \in X_i, \mathrm{Orbcount}(G, S) \neq C\}$. If one of these sets is empty, X_i is not split.

4.2. Algorithm

We will now present our algorithm. First, we give a technical definition which will be used in proving the correctness of our algorithm.

Definition 4.6. For all $n \in \mathbb{N}$ we define $\mathcal{L}_n$ to be the set of lists of length n whose entries are non-empty subsets of Ω. If $X \in \mathcal{L}_n$, then as a convention we write $X_1, \dots, X_n$ for the entries of the list X.

If $X \in \mathcal{L}_n$ and $H \leqslant G \leqslant S_n$ is such that $|G : H| = k \in \mathbb{N}$ and $Q = \{q_1, \dots, q_k\}$ is a set of coset representatives of H in G, then we define X^Q to be the list whose first k entries are $X_1^{q_1}, \dots, X_1^{q_k}$, followed by $X_2^{q_1}, \dots, X_2^{q_k}$ until the last k entries are $X_n^{q_1}, \dots, X_n^{q_k}$. We note that $X^Q \in \mathcal{L}_{n \cdot k}$.

Let $X, Y \in \mathcal{L}_n$. We say that X and Y are G-equivalent if and only if there exist a permutation σ of $\{1, \dots, n\}$ and group elements $g_1, \dots, g_n \in G$ such that, for all $i \in \{1, \dots, n\}$, it holds that $Y_i = X_{i^\sigma}^{g_i}$.

We now prove a series of three lemmas about coset representatives, which form the basis for the correctness proof of our algorithm. They are used to perform the recursive step, moving from a group to a subgroup.

Lemma 4.7. *Suppose that G is a permutation group on a set Ω, that H is a subgroup of G of index $k \in \mathbb{N}$ and that T is a set of left coset representatives of H in G.*

Then the following are true:

(i) $|T| = k$.
(ii) *If $T = \{t_1, \dots, t_k\}$ and $g \in G$ and if, for all $i \in \{1, \dots, k\}$, we define $q_i := gt_i$, then $Q := \{q_1, \dots, q_k\}$ is also a set of coset representatives of H in G. In particular there is a bijection from Q to any set of left coset representatives of H in G.*

Proof. By definition the index of H in G is the number of (left or right) cosets of H in G.

For the second statement we let $i, j \in \{1, \dots, k\}$ be such that $q_i H = q_j H$, hence $gt_i H = gt_j H$. Then $t_j^{-1} t_i = t_j^{-1} g^{-1} g t_i \in H$ and hence $t_i H = t_j H$. Hence $i = j$ because t_i and t_j are from a set of coset representatives. □

Lemma 4.8. *Suppose that G is a permutation group on a set Ω and that $S, T \subseteq \Omega$ are such that the lists $[S]$ and $[T]$ are G-equivalent.*

Let H be a subgroup of G of index $k \in \mathbb{N}$ and let $P = \{p_1, \dots, p_k\}$ and $Q := \{q_1, \dots, q_k\}$ be sets of left coset representatives of H in G.

Then $[S]^P$ and $[T]^Q$ are H-equivalent.

Proof. As $[S]$ and $[T]$ are G-equivalent, we know that there exists a group element $g \in G$ such that $S^g = T$.

We fix g, for all $i \in \{1, \dots, k\}$ we let $t_i := gq_i$ and we consider the set $T := \{t_i \mid i \in \{1, \dots, k\}\}$. Then T is also a set of left coset representatives of H in G, by Lemma 4.7. As P is also a set of left coset representatives, we know that T and P have the same size, so there is a bijection from P to T. This can be expressed in the following way:

There is a permutation $\sigma \in S_k$ such that, for all $i \in \{1, .., k\}$, it is true that $p_{i^\sigma} H = t_i H$. That means there is a unique $h_i \in H$ such that $p_{i^\sigma} h_i = t_i$.

Let now $S_i := S^{p_i}$ and $T_i := T^{q_i}$ for $i \in \{1, \dots, k\}$, then

$$T_i = T^{q_i} = (S^g)^{q_i} = S^{t_i} = S^{p_{i^\sigma} h_i} = (S^{p_{i^\sigma}})^{h_i} = (S_{i^\sigma})^{h_i},$$

hence $[S]^P$ and $[T]^Q$ are H-equivalent. □

Lemma 4.9. *Suppose that G is a permutation group on a set Ω, that $n \in \mathbb{N}$ and that $X, Y \in \mathcal{L}_n$ are G-equivalent. Let H be a subgroup of G of index $k \in \mathbb{N}$ and let $P = \{p_1, \dots, p_k\}$ and $Q := \{q_1, \dots, q_k\}$ be sets of left coset representatives of H in G.*

Then X^P and Y^Q are H-equivalent.

Proof. As X and Y are G-equivalent, we know that there exist a permutation $\sigma \in S_n$ and $g_1, \dots, g_n \in G$ such that $Y_i = X_{i^\sigma}^{g_i}$ for all $i \in \{1, \dots, n\}$. We fix this permutation σ.

If $i \in \{1, \dots, n\}$, then $[X_{i^\sigma}]$ and $[Y_i]$ satisfy the hypothesis of Lemma 4.8, so it follows that $[X_{i^\sigma}]^P$ and $[Y_i]^Q$ are H-equivalent.

So we find a permutation $\alpha_i \in S_k$ and group elements $h_{i1}, \dots, h_{ik} \in H$ such that $(X_{i^\sigma}^{p_{j^{\alpha_i}}})^{h_{ij}} = Y_i^{q_i}$ for all $j \in \{1, \dots, k\}$.

Using σ and $\alpha_1, \dots, \alpha_k$ we define a permutation γ on $1, \dots, n \cdot k$.

First we express $l \in \{1, \dots, n \cdot k\}$ uniquely as $l = c_l \cdot k + r_l$ where $c_l \in \{0, \dots, n-1\}$ and $r_l \in \{1, \dots, k\}$ and we define

$$l^\gamma := (c_l + 1)^\sigma \cdot k + r_l^{\alpha_{c_l+1}}.$$

This is well-defined because of the ranges of c_l and r_l and it is a permutation because of the uniqueness of the expression and because σ and $\alpha_1, \dots, \alpha_n$ are permutations.

Then, for each $l \in \{1, \dots, n \cdot k\}$, expressed as $l = c_l \cdot k + r_l$ as we did above, we set $h := h_{c_l+1, r_l}$, $X'_l := X_{c_l+1}^{p_{r_l}}$ and $Y'_l := Y_{c_l+1}^{q_{r_l}}$.

Then $X^P = [X'_1, \dots, X'_{n \cdot k}]$ and $Y^Q = [Y'_1, \dots, Y'_{n \cdot k}]$.

If we set $p := p_{r_l^{\alpha_{c_l+1}}}$ and $q := q_{r_l}$, then we have, for all $l = c_l \cdot k + r_l$, that

$$X'^h_{l^\gamma} = (X^p_{(c_l+1)^\sigma})^h = ((X^\sigma_{c_l+1})^p)^h = Y^q_{c_l+1} = Y'_l.$$

This is H-equivalence. □

We can now describe the algorithm we use to compute canonical images, and prove that it works correctly.

Definition 4.10. Suppose that Ω is a finite set, that G is a permutation group on Ω, that $L \in \mathcal{L}_k$ and that P is an ordered partition on $\mathcal{P}(\Omega)$.

- An Ω-*selector* is a function $\mathcal{S}$ such that
  - $\mathcal{S}(\Omega, G, L, P) = \omega \in \Omega$, where $|\omega^G| > 1$;
  - $\mathcal{S}(\Omega, G, L, P) = \mathcal{S}(\Omega, G, M, P)$ whenever L and M from $\mathcal{L}_k$ are G-equivalent.
- An *Ordering refiner* is a function $\mathcal{O}$ such that for all G-invariant partitions P of $\mathcal{P}(\Omega)$
  - $\mathcal{O}(\Omega, G, L, P) = P'$, where P' is a G-invariant refinement of P;
  - $\mathcal{O}(\Omega, G, L, P) = \mathcal{O}(\Omega, G, M, P)$ whenever L and M from $\mathcal{L}_k$ are G-equivalent.

An ordering refiner cannot return a total ordering, unless G acts trivially, because the partial ordering cannot distinguish between values that are contained in the same orbit of G.

Our method for finding canonical images is outlined in Algorithm 2. It recursively searches for the minimal image of a collection of lists, refining the ordering that is used as search progresses.

Algorithm 2 CanImage.

Require: $\mathcal{S}$ is an Ω-selector, $\mathcal{O}$ is an ordering refiner

procedure CanImageRecurse(Ω, G, L, P)
    if $|G| = 1$ **then**
        P' := Standard completion of P
        return Smallest member of L under P'
    $H := G_{\mathcal{S}(\Omega, G, L, P)}$
    Q := coset representatives of H in G
    $P' := \mathcal{O}(\Omega, H, L^Q, P)$
    $L' := [S \mid S \in L^Q, \nexists T \in L^Q . T <_{P'} S]$
    return CanImageRecurse(Ω, H, L', P')

procedure CanImageBase($\Omega, InputG, InputS$)
    return CanImageRecurse($\Omega, InputG, [InputS], [\mathcal{P}(\Omega)]$)

Theorem 4.11. *Suppose that Ω is a finite set, that G is a permutation group on Ω, and that $X \subseteq \Omega$. Then* CanImageBase$(\Omega, G, X) \in X^G$.

Proof. In every step of Algorithm 2 the list of considered sets is a list of elements of X^G. $\square$

Theorem 4.12. *Let Ω be a finite set and G a permutation group on Ω. Let $X, Y \in \mathcal{L}_k$ be G-equivalent, and let P be a G-invariant ordered partition of $\mathcal{P}(\Omega)$. Then*

$$\textsc{CanImage}(\Omega, G, X, P) = \textsc{CanImage}(\Omega, G, Y, P).$$

Proof. We proceed by induction on the size of G.

The base case is $|G| = 1$. As X and Y are G-equivalent, X and Y contain the same sets, possibly in a different order. For a given P and Ω, there is only one standard completion of P that gives a complete ordering on $\mathcal{P}(\Omega)$, and so X and Y have the same smallest element under the standard completion of P and so the claim follows.

Consider now any non-trivial group G, and suppose for our induction hypothesis that the claim holds for all groups H where $|H| < |G|$.

Definition 4.10 and the fact that $\mathcal{S}$ is an Ω-selector imply that

$$H = G_{\mathcal{S}(\Omega,G,X,P)} = G_{\mathcal{S}(\Omega,G,Y,P)}.$$

Moreover, H is a proper subgroup of G. We take two sets Q_1 and Q_2 of coset representatives of H in G, which are not necessarily equal.

Since $\mathcal{O}$ is an ordering refiner, it holds that

$$P' = \mathcal{O}(\Omega, H, X, P) = \mathcal{O}(\Omega, H, Y, P)$$

By Lemma 4.9, X^{Q_1} and Y^{Q_2} are H-equivalent, and by definition P' is G-invariant. If we identify the cell P'_i of P' that contains the smallest element of X^{Q_1}, then L'_X contains those elements of X^{Q_1} that are in P'_i. Each of these elements is H-equivalent to an element of Y^{Q_2}, and therefore L'_X is H-equivalent to L'_Y. Then the induction hypothesis yields that

$$\textsc{CanImage}(\Omega, H, X', P) = \textsc{CanImage}(\Omega, G, Y', P),$$

so the claim follows by induction. □

We see from Theorem 4.11 and 4.12 that

$$\textsc{CanImage}(\Omega, G, X, P) = \textsc{CanImage}(\Omega, G, Y, P) \text{ if and only if } Y \in X^G.$$

We note that Algorithm 2 can easily be adapted to return an element g of G such that $X^g = \textsc{CanImage}(\Omega, G, X, P)$. This happens by attaching to each set, when it is created, the permutation that maps it to the original input S. We omit this addition for readability.

5. Experiments

In this section, we will compare how well our new algorithms perform in comparison with the **MinImage** function of Linton's.

All of our experiments are performed in GAP 4.8.8 [3]. The implementation of our algorithm which we test is provided by the `CanonicalImage` function in version 1.1.0 of the `Images` GAP package [7]. The orderings we describe are all provided by `Images`, with the same names. The `Images` GAP package requires the `Ferret` GAP package [6] – we used version 1.0.0 of `Ferret` for the experiments in this paper.

We consider a selection of different canonical image algorithms and we analyze how they perform compared to each other, and compared to the traditional minimal image algorithm of Linton's, which we will refer to as **MinImage**.

The first three algorithms that we consider come from Section 3. They produce, given a group G on a set Ω, an ordering of Ω. This ordering is then used in **MinImage**.

(i) **FixedMinOrbit** uses results from Section 3 to calculate an alternative ordering of Ω, choosing small orbits first.
(ii) **FixedMaxOrbit** works similarly to **FixedMinOrbit**, choosing large orbits first.

We also consider algorithms that dynamically choose which value to branch on as search progresses. We will use the following lemma for the proof of correctness for all our orderings.

Lemma 5.1. *Let Ω be a finite set, G a permutation group on Ω, and P an ordered partition of $\mathcal{P}(\Omega)$.*

Let $L \in \mathcal{L}_n$, and $\text{Count} = [\text{Orbcount}(G, S) \mid S \in L]$.

- *Any function that accepts (Ω, G, L, P) and returns some $\omega \in \Omega$ with $|\omega^G| > 1$ that is invariant under reordering the elements of* Count*, is an Ω-selector.*
- *Any function that accepts (Ω, G, L, P) and returns either P or the point refinement of P by some ω^G for $\omega \in \Omega$, and is invariant under permutation of the elements of* Count*, is an ordering refiner.*
- *Any function that accepts (Ω, G, L, P) and returns either P or the orbit refiner of P by some member of* Count *and is invariant under permutation of the elements of* Count*, is an ordering refiner.*

Proof. The only thing we have to show is that, if L and M are G-equivalent, then any of the functions above yield the same result for inputs (Ω, G, L, P) and (Ω, G, M, P). By Lemma 4.4, any G-equivalent lists L and M will produce the same list Count, up to reordering of elements, and hence the claim follows. □

Firstly we define a list of orderings. Each of these orderings chooses an orbit, or list of orbits, to branch on – we will then make an Ω-selector by choosing the smallest element in any of the orbits selected, to break ties (we could choose any point, as long as we picked it consistently). Each of these algorithms operates on a list $L \in \mathcal{L}_k$. In each case we look for an orbit, ignoring orbits of size one (as fixing a point that was already fixed leads to the same group).

Firstly we will consider two algorithms that only consider the group, and not L:

(i) **MinOrbit** Choose a point from a shortest non-trivial orbit that has a non-empty intersection with at least one element of L.

(ii) **MaxOrbit** Choose a point from a longest non-trivial orbit that has a non-empty intersection with at least one element of L.

We also consider four algorithms that consider both the group, and L.

In the following, for an orbit o

(i) **RareOrbit** minimises $\sum_{s \in L} |s \cap o|$,
(ii) **CommonOrbit** maximises $\sum_{s \in L} |s \cap o|$,
(iii) **RareRatioOrbit** minimises $\log(\sum_{s \in L} |s \cap o|)/|o|$,
(iv) **CommonRatioOrbit** maximises $\log(\sum_{s \in L} |s \cap o|)/|o|$.

The motivation for **RareOrbit** is that this is the branch which will lead to the smallest size of the next level of search – this exactly estimates the size of the next level if our ordering refiner only fixed a point in orbit o. We therefore expect, conversely, **CommonOrbit** to perform badly and to produce very large searches.

One limitation of **RareOrbit** is that it will favour smaller orbits, as these will tend to produce few sets at the next level – whereas we want to minimize the size of the whole search. As an example, if we have two levels of search where we split on an orbit of size two and each time create 10 times more sets, this is equivalent to splitting once on an orbit of size four and creating 100 times more sets. **RareRatioOrbit** aims to minimise the growth in the number of sets, taking into account the size of the orbit. **CommonRatioOrbit** is the reverse of this heuristic, so we expect it to perform badly.

For each of these orderings, we use the ordering refiner that takes each fixed point of G in their order in Ω, and performs a point refinement by each recursively in turn. By repeated application of Lemma 4.4, this is a G-invariant ordering refiner.

We also have a set of orderings which make use of orbit counting. To keep the number of experiments under control, we used the **RareOrbit** strategy in each case to choose which point to branch on next, and we also build an orbit refiner.

Given an unordered list of orbit counts,

(i) **RareOrbitPlusMin** chooses the lexicographically smallest one.
(ii) **RareOrbitPlusRare** chooses the least frequently occurring orbit count list (using the lexicographically smallest to break ties).
(iii) **RareOrbitPlusCommon** chooses the most frequently occurring orbit count list (using the lexicographically smallest to break ties).

5.1. Experiments

In this section we perform a practical comparison of our algorithms, and the **MinImage** algorithm of Linton, for three different families of problems: grid groups, m-sets, and primitive groups.

5.2. Experimental design

We will consider three sets of benchmarks in our testing. In each experiment, given a permutation group that acts on $\{1, \ldots, n\}$, we will run an experiment with each of our orderings to find the canonical image of a set of size $\lfloor \frac{n}{2} \rfloor$, $\lfloor \frac{n}{4} \rfloor$ and $\lfloor \frac{n}{8} \rfloor$.

We run our algorithms on a randomly chosen conjugate of each primitive group, to randomize the initial ordering of the integers the group is defined over. The same conjugate is used of each group in all experiments, and when choosing a random subset of size x from a set S, we always choose the same random subset. We use a timeout of five minutes for each experiment. We force GAP to build a stabilizer chain for each of our groups before we begin our algorithm, because this can in some cases take a long time.

For each size of set and each ordering, we measure three things. The total number of problems solved, the total time taken to solve all problems, counting timeouts as 5 minutes, and the number of moved points of the largest group solved. Our experiments were all performed on an Intel(R) Xeon(R) CPU E5-2640 v4 running at 2.40 GHz, with twenty cores. Each copy of GAP was allowed a maximum of 6 GB of RAM.

5.2.1. Grid groups

In this experiment, we look for canonical images of sets in grid groups.

Definition 5.2. Let $n \in \mathbb{N}$. The direct product $S_n \times S_n$ acts on the set $\{1, \ldots, n\} \times \{1, \ldots, n\}$ of pairs in the following way:

For all $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ and all $(\sigma, \tau) \in S_n \times S_n$ we define

$$(i, j)^{(\sigma,\tau)} := (i^\sigma, j^\tau).$$

The subgroup $G \leqslant Sym(\{1, \ldots, n\} \times \{1, \ldots, n\})$ defined by this action is called the $n \times n$ **grid group**.

We note that, while the construction of the grid group is done by starting with an n by n grid of points and permuting rows or columns independently of each other, we actually represent this group as a subgroup of $S_{n \cdot n}$, and we do not assume prior knowledge of the grid structure of the action.

We ran experiments on the grid groups for grids of size 5×5 to 100×100. The results of this experiment are given in Table 1.

Table 1
Finding canonical images in grid groups.

	Stab	$\lfloor \frac{n}{2} \rfloor$			$\lfloor \frac{n}{4} \rfloor$			$\lfloor \frac{n}{8} \rfloor$		
	Search	# solved	largest	time	# solved	largest	time	# solved	largest	time
RareOrbitPlusRare	F	56	4,225	12,549	88	8,464	10,474	91	9,216	13,663
RareOrbitPlusMin	F	54	4,225	13,207	89	9,604	10,546	90	9,025	13,484
RareOrbitPlusCommon	F	30	2,209	19,306	65	5,476	15,408	90	9,216	14,009
SingleMaxOrbit	F	11	225	24,671	27	961	25,795	52	3,136	23,532
RareOrbit	F	8	144	24,763	17	625	27,213	34	4,096	25,979
MinOrbit	F	8	144	24,739	16	400	27,735	31	1,521	26,706
CommonRatioOrbit	F	7	121	24,990	15	400	28,255	29	1,296	27,322
FixedMinOrbit	T	8	144	24,796	13	361	28,619	22	841	28,841
FixedMinOrbit	F	8	144	24,786	13	361	28,573	21	841	28,859
MinImage	F	4	64	25,819	8	144	30,039	10	196	31,851
MinImage	T	4	64	25,822	7	144	29,962	10	196	31,829
RareRatioOrbit	F	3	49	26,126	5	81	30,379	8	144	32,804
MaxOrbit	F	3	49	26,123	5	81	30,373	8	144	32,771
FixedMaxOrbit	F	3	49	26,120	5	81	30,374	8	144	32,725
FixedMaxOrbit	T	3	49	26,113	5	81	30,381	8	144	32,450
CommonOrbit	F	3	49	26,131	5	81	30,396	8	144	32,826

The basic algorithm, **MinImage**, is only able to solve 22 problems within the timeout. **FixedMinOrbit** solves 43 problems, while being implemented as a simple pre-processing step to **MinImage**. The dynamic **MinOrbit** is able to solve 55 problems, and the best orbit-based strategy, **SingleMaxOrbit**, solves 70 problems. However the advanced techniques, which filter by orbit lists, perform much better. Even ordering by the most common orbit list leads to solving over 185 problems, and the best strategy, **RareOrbitPlusRare**, solves 235 out of the total 288 problems.

Furthermore, for these large groups, the algorithms are still performing very small searches: For example **FixedMinOrbit**, on its largest solved problem with size $\lfloor \frac{n}{2} \rfloor$ sets and a grid of size 12×12, generates $793,124$ search nodes, while **RareOrbitPlusMin** produces only $183,579$ search nodes on its largest solved problem with $\lfloor \frac{n}{2} \rfloor$ sets (65×65).

5.2.2. M-sets

Linton [10] considers, given integers n and m, defining a permutation group on the set T of all subsets of size m of $\{1, \dots, n\}$ under the action on T of S_n acting on the members of the m-sets. He then looks for minimal images of randomly chosen subsets of T of size k, under the standard lexicographic ordering on sets.

We ran experiments for $m = 2$ and $n \in \{10, 15, \dots, 100\}$, for $m = 4$ and $n \in \{10, 15, \dots, 35\}$, for $m = 6$ and $n \in \{10, 15, 20\}$ and finally $m = 8$, $n = 10$ as described in Section 5.2. We choose these 30 experiments as these were the problems that any of our techniques were able to solve in under 5 minutes. The results of our experiments are shown in Table 2.

Similarly to our experiments on grid groups, we find that the standard **MinImage** algorithm is only able to solve a very small set of benchmarks. Some of the better algorithms, including **FixedMinOrbit**, are able to solve 9 problems. In particular, once again **MinOrbit** is not significantly better than **FixedMinOrbit**, although it is slightly faster on average over all problems.

However, the orbit-based strategies do much better, solving all the problems which we set. In the case of sets that contain an eighth of all m-sets, the best technique is able to solve all problems any technique can solve, in under 5 minutes. The largest solved problem, which was instance $n = 35, m = 4$ for a set on an eighth of all m-sets, is solved in only $6,594$ search nodes by **RareOrbitPlusMin**, while the largest solved problem of **MinImage**, $n = 15$, $m = 4$ takes $631,144$ search nodes.

5.2.3. Comparison to graph canonical image

A set of 2-sets can be viewed as an undirected graph, where the two sets represent the edges. The problem of finding the canonical image of this set of 2-sets is equivalent to the traditional problem of finding a canonical image of this graph. We can therefore perform a comparison between our technique and Nauty, for these problems. Nauty is able to a find canonical image for all our 2-set problems almost instantly. We investigated why Nauty was able to outperform us by such a large margin, and found three problems. We list the most important one first.

Table 2
Finding canonical images in M-set groups.

	Stab	$\lfloor \frac{n}{2} \rfloor$			$\lfloor \frac{n}{4} \rfloor$			$\lfloor \frac{n}{8} \rfloor$		
	Search	# solved	largest	time	# solved	largest	time	# solved	largest	time
RareOrbitPlusMin	F	28	27,405	1,290	30	52,360	835	30	52,360	251
RareOrbitPlusRare	F	27	12,650	1,235	30	52,360	1,107	30	52,360	322
RareOrbitPlusCommon	F	27	12,650	1,819	30	52,360	1,189	30	52,360	335
MinOrbit	F	9	1,365	6,701	22	6,435	4,143	28	27,405	1,203
RareOrbit	F	9	1,365	6,769	21	6,435	4,190	28	27,405	1,276
FixedMinOrbit	F	9	1,365	6,671	22	6,435	4,292	27	12,650	1,765
FixedMinOrbit	T	9	1,365	6,678	22	6,435	4,291	27	12,650	1,766
CommonRatioOrbit	F	9	1,365	6,782	21	6,435	4,215	28	27,405	1,348
MinImage	F	4	210	7,866	5	210	7,562	7	1,365	7,033
MinImage	T	4	210	7,843	5	210	7,573	7	1,365	7,037
SingleMaxOrbit	F	4	210	7,813	5	210	7,554	6	1,365	7,325
FixedMaxOrbit	T	4	210	7,853	5	210	7,691	6	210	7,569
RareRatioOrbit	F	4	210	7,923	5	210	7,642	5	210	7,500
MaxOrbit	F	4	210	7,915	5	210	7,695	5	210	7,500
FixedMaxOrbit	F	4	210	7,890	5	210	7,666	5	210	7,500
CommonOrbit	F	4	210	7,947	5	210	7,724	5	210	7,500

Table 3
Finding canonical images in primitive groups.

	Stab	$\lfloor \frac{n}{2} \rfloor$		$\lfloor \frac{n}{4} \rfloor$		$\lfloor \frac{n}{8} \rfloor$	
	Search	# solved	time	# solved	time	# solved	time
RareOrbitPlusMin	F	5,689	64,983	5,749	39,564	5,840	10,287
RareOrbitPlusRare	F	5,656	81,814	5,738	43,202	5,825	19,025
RareOrbitPlusCommon	F	5,561	113,011	5,721	59,740	5,816	25,392
FixedMinOrbit	T	5,360	220,076	5,623	82,684	5,817	18,295
FixedMinOrbit	F	5,354	251,786	5,628	99,253	5,816	30,193
MinOrbit	F	5,348	263,053	5,641	104,241	5,844	27,720
RareOrbit	F	5,324	272,631	5,632	105,537	5,844	29,365
CommonRatioOrbit	F	5,323	277,050	5,629	107,561	5,844	32,123
SingleMaxOrbit	F	4,811	465,908	5,250	253,467	5,648	112,147
MinImage	T	4,723	390,334	5,163	242,477	5,631	88,048
MinImage	F	4,710	501,952	5,180	280,686	5,633	106,983
FixedMaxOrbit	F	4,659	514,618	5,119	298,321	5,587	132,508
FixedMaxOrbit	T	4,674	392,753	5,095	251,001	5,583	107,433
MaxOrbit	F	4,641	544,222	5,104	310,402	5,587	144,182
RareRatioOrbit	F	4,614	559,690	5,090	316,609	5,586	152,201
CommonOrbit	F	4,604	569,047	5,086	319,288	5,586	154,142

- The central algorithm of Nauty makes use of properties of the form "vertices with i neighbours can only map to other vertices with i neighbours". Our algorithm does not make use of this property, as it represents a much more complex condition when considered on m-sets. Further, while we could add a special case specifically for when the group we are considering is the symmetric group operating on m-sets, we would prefer to find a more general technique.
- Our algorithm spends a large proportion of its time calculating stabilizer chains, and mapping sets through elements of the group. This is not required for the graphs.
- Our algorithm is written in GAP rather than highly optimized C.

The most important results to draw from this comparison is that our algorithms should not be viewed as a replacement for graph isomorphism algorithms. We are investigating how to close this performance gap, without special casing.

5.2.4. Primitive groups

In this experiment we look for canonical images of sets under the action of primitive groups which move between 2 and $1,000$ points. We remove the natural alternating and symmetric groups, as finding minimal and canonical images in these groups is trivial and can be easily special-cased. So we look at a total number of $5,845$ groups, each of which was successfully treated by at least one algorithm.

We perform the experiment as described in Section 5.2. The results are given in Table 3.

All algorithms are able to solve a large number of problems. This is unsurprising as many primitive groups are quite small (for example the cyclic groups), so any technique is able to brute-force search many problems. However, we can still see that, for the hardest problems $\lfloor \frac{n}{2} \rfloor$, many algorithms outperform **MinImage**, and the techniques that use extra orbit-counting filtering solve 300 more problems, and they run much faster.

For the easiest set of problems, $\lfloor \frac{n}{8} \rfloor$, we see that the algorithm **RareOrbitPlusMin**, which usually performs best, solves slightly fewer problems. This is because there are a small number of groups where the extra filtering provides no search reduction, but still requires a small overhead in time. However, the total time taken is still much smaller, and the algorithm only fails to solve five problems. These five problems involve groups that are isomorphic to the affine general linear groups AGL(8, 2), AGL(6, 3) and AGL(9, 2), and the projective linear group PSL(9, 2). This suggests that the linear groups may be a source of hard problems for canonical image algorithms in the future.

5.2.5. Experimental conclusions

Our experiments show that using **FixedMinOrbit** is almost always superior to **MinImage**. As implementing **FixedMinOrbit** requires a fairly small amount of code and time over **MinImage**, this suggests that any implementations of Linton's algorithm should have **FixedMinOrbit** added, because this provides a substantial performance boost, for relatively little extra coding.

Algorithms that dynamically order the underlying set, such as **MinOrbit** and **RareOrbit** provide only a small benefit over **FixedMinOrbit**. Algorithms which add orbit counting provide a much bigger gain, often allowing solving problems on groups many orders of magnitude larger than before, thereby greatly advancing the state of the art.

6. Conclusions

We present a general framework and a new set of algorithms for finding the canonical image of a set under the action of a permutation group. Our experiments show that our new algorithms outperform the previous state of the art, often by orders of magnitude.

Our basic framework runs on the concept of refiners and selectors and is not limited to finding only canonical images of subsets of Ω. In future work we will investigate families of refiners and selectors that allow finding canonical images for many other combinatorial objects.

Acknowledgments

All authors thank the DFG (**Wa 3089/6-1**) and the EPSRC (**EP/M022641/1**) for supporting this work. The first author would like to thank the Royal Society, and the EPSRC (**EP/M003728/1**). The third author would like to acknowledge support from the OpenDreamKit Horizon 2020 European Research Infrastructures Project (#676541). The first and third author thank the Algebra group at the Martin-Luther Universität Halle-Wittenberg for the hospitality and the inspiring environment. The fourth author wishes to thank the Computer Science Department of the University of St Andrews for its hospitality during numerous visits.

References

[1] P.T. Darga, K.A. Sakallah, I.L. Markov, Faster symmetry discovery using sparsity of symmetries, in: 2008 45th ACM/IEEE Design Automation Conference, June 2008, pp. 149–154.
[2] Andreas Distler, Chris Jefferson, Tom Kelsey, Lars Kotthoff, The Semigroups of Order 10, Springer, Berlin, Heidelberg, 2012, pp. 883–899.
[3] The GAP Group, GAP – groups, algorithms, and programming, version 4.8.8, http://www.gap-system.org, 2017.
[4] Ian P. Gent, Barbara M. Smith, Symmetry breaking in constraint programming, in: Proceedings of the 14th European Conference on Artificial Intelligence, IOS Press, 2000, pp. 599–603.
[5] Derek F. Holt, Bettina Eick, Eamonn A. O'Brien, Handbook of Computational Group Theory, Discrete Mathematics and Its Applications, Chapman & Hall/CRC, Boca Raton, 2005.
[6] Christopher Jefferson, Ferret – GAP package, version 1.0.0, https://gap-packages.github.io/ferret/, 2018.
[7] Christopher Jefferson, Eliza Jonauskyte, Markus Pfeiffer, Markus Waldecker, Images – GAP package, version 1.1.0, https://gap-packages.github.io/images/, 2018.
[8] Tommi Junttila, Petteri Kaski, Engineering an efficient canonical labeling tool for large and sparse graphs, in: David Applegate, Gerth Stølting Brodal, Daniel Panario, Robert Sedgewick (Eds.), Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics, SIAM, 2007, pp. 135–149.
[9] Jeffrey S. Leon, Permutation group algorithms based on partitions, I: theory and algorithms, J. Symbolic Comput. 12 (4) (1991) 533–583.
[10] Steve Linton, Finding the smallest image of a set, in: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, ISSAC '04, ACM, New York, NY, USA, 2004, pp. 229–234.
[11] Brendan D. McKay, Practical graph isomorphism, Congr. Numer. (30) (1980) 45–87.
[12] Brendan D. McKay, Adolfo Piperno, Practical graph isomorphism, II, J. Symbolic Comput. 60 (2014) 94–112.
[13] Christian Pech, Sven Reichard, Enumerating Set Orbits, Springer, Berlin, Heidelberg, 2009, pp. 137–150.
[14] Leonard H. Soicher, On the structure and classification of somas: generalizations of mutually orthogonal latin squares, Electron. J. Combin. 6 (1999).