Journal of Symbolic Computation 114 (2023) 18–36

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Perfect refiners for permutation group backtracking algorithms

Christopher Jefferson, Rebecca Waldecker, Wilf A. Wilson

ARTICLE INFO

Article history:
Received 18 December 2021
Received in revised form 8 April 2022
Accepted 12 April 2022
Available online 25 April 2022

Keywords:
Permutation groups
Backtrack search
Search algorithms
Refiners

ABSTRACT

Backtrack search is a fundamental technique for computing with finite permutation groups, which has been formulated in terms of points, ordered partitions, and graphs. We provide a framework for discussing the most common forms of backtrack search in a generic way. We introduce the concept of perfect refiners to better understand and compare the pruning power available in these different settings. We also present a new formulation of backtrack search, which allows the use of graphs with additional vertices, and which is implemented in the software package Vole. For each setting, we classify the groups and cosets for which there exist perfect refiners. Moreover, we describe perfect refiners for many naturally-occurring examples of stabilisers and transporter sets, including applications to normaliser and subgroup conjugacy problems for 2-closed groups.

1. Introduction

A careful backtrack search through the elements of a symmetric group is the fastest general purpose technique, in practice, for solving many computational problems in finite permutation groups. This includes finding normalisers and intersections of subgroups, and stabilisers of sets and graphs.

At the heart of these backtracking algorithms are refiners. Refiners are functions which are used to prune redundant parts of the search. They provide the main facility for taking into account the structural aspects of the problem at hand, and making clever deductions. Better refiners are more likely to prune more efficiently, and can reduce search times by orders of magnitude. Refiners, therefore, are

E-mail address: rebecca.waldecker@mathematik.uni-halle.de (R. Waldecker).

https://doi.org/10.1016/j.jsc.2022.04.007

an obvious and ongoing target of further research (see for example Jefferson et al., 2019a for recent work in this area), and they are the focus of this article.

Roughly speaking, backtracking algorithms organise permutations as bijective maps on sets of combinatorial structures. The first version was originally formulated by Sims, organised around a base and strong generating set (see Sims, 1971). Two decades later, Leon reformulated this in terms of ordered partitions (Leon, 1991, 1997), and thereby obtained significantly improved performance in many cases. This technique was enhanced with orbital graphs in Theißen (1997), Jefferson et al. (2019a). More recently, this use of orbital graphs inspired a reformulation in Jefferson et al. (2021) around labelled digraphs.

The motivation for organising a search around more sophisticated structures is to allow some sets of permutations to be represented with greater fidelity, so that refiners can prune more effectively. However, when choosing the appropriate search infrastructure for a particular problem, we need to keep in mind that more complicated structures do not always offer better pruning, that they are more expensive to compute with, and that it is easy to find relatively small cases where *all* existing backtracking algorithms exhibit poor performance. This holds especially for subgroup conjugacy and normaliser problems. It is therefore important to develop tools for understanding which problems are well suited to which kinds of backtrack search and refiners, and to understand the limitations of the existing techniques, with the aim of developing improvements.

This article has three main purposes. Firstly, we introduce the concept of perfect refiners, which are those with maximal pruning power within a given search framework. This concept is also meant to initiate a discussion of the quality of refiners, both within and across various backtrack search techniques. Secondly, we describe an extension to the graph backtracking technique that permits the use of graphs with additional vertices, and which enables many more perfect refiners than previous techniques. There already is an implementation available, in Vole (Chang et al., 2021). Finally, we partially classify, and give examples of, perfect refiners in each setting of backtrack search.

Our results show that graph backtracking and extended graph backtracking admit perfect refiners for many natural problems. This goes some way to explaining the experimental data in Jefferson et al. (2021, Section 9), which demonstrated that many problems could be solved without actual backtracking happening during the search. Furthermore, we find that extended backtracking enables improved refiners for normaliser and conjugacy problems. In particular, we give refiners for normalisers and conjugacy within the extended graph backtracking framework that are perfect in some cases.

This article is organised as follows. In Section 2, we present some necessary background definitions and notation; in particular, we briefly describe backtrack search in finite symmetric groups. We then give definitions and results about arbitrary refiners in Section 3 and perfect refiners in Section 4. In Section 5, we examine perfect refiners for stabiliser and transporter problems. In Section 6, we introduce extended graph backtracking. In Section 7, we compare the various kinds of backtrack search and their potentials for perfect refinement. In Section 8, we give examples of perfect refiners for many natural problems. We conclude with some final remarks in Section 9.

Acknowledgements The results discussed here build on work in Jefferson et al. (2021), and we include elements of an earlier draft of that article (Jefferson et al., 2019, Section 5.1). We therefore thank the VolkswagenStiftung (**Grant no. 93764**) and the Royal Society (**Grant code URF\R\180015**) again for their financial support of this earlier work. For financial support during the more recent advances, we thank the DFG (**Grant no. WA 3089/9-1**) and again the Royal Society (**Grant codes RGF\EA\181005** and **URF\R\180015**).

2. Background and notation

In this section, we give some notation and definitions, and we introduce background concepts including backtrack search in the symmetric group. Our terminology and notation closely follows that of Jefferson et al. (2021).

Sets and lists feature prominently in this article: a set is an unordered duplicate-free collection of objects, whereas a list has an ordering, and may contain duplicates. If L and K are lists, then

$|L|$ denotes the number of elements in L, $L\|K$ denotes the concatenation of L and K, and if $i \in \{1, \dots, |L|\}$, then $L[i]$ denotes the element of L in the i-th position.

Throughout this article, Ω is a nonempty finite set, and $\mathrm{Sym}(\Omega)$ is the symmetric group on Ω. We denote any action of $\mathrm{Sym}(\Omega)$ by exponentiation. Given an action of $\mathrm{Sym}(\Omega)$ on a set $\mathcal{O}$, we iteratively define an action of $\mathrm{Sym}(\Omega)$ induced on the set of all subsets of $\mathcal{O}$, and on the set of all finite lists with elements in $\mathcal{O}$. In more detail, for all $x_1, \dots, x_k \in \mathcal{O}$ and $g \in \mathrm{Sym}(\Omega)$, we define $\{x_1, \dots, x_k\}^g := \{x_1^g, \dots, x_k^g\}$ and $[x_1, \dots, x_k]^g := [x_1^g, \dots, x_k^g]$. For example, if $\Omega := \{1, 2, 3, 4\}$, then the image of the set-of-lists $\{[1, 2], [2, 3], [3, 2]\}$ under the permutation $(1\,2)(3\,4)$ is

$$\begin{aligned}\{[1, 2], [2, 3], [3, 2]\}^{(1\,2)(3\,4)} &= \{[1, 2]^{(1\,2)(3\,4)}, [2, 3]^{(1\,2)(3\,4)}, [3, 2]^{(1\,2)(3\,4)}\}\\ &= \{[2, 1], [1, 4], [4, 1]\}.\end{aligned}$$

Let $\mathcal{O}$ be a set on which $\mathrm{Sym}(\Omega)$ acts. We define $\mathrm{Stacks}(\mathcal{O})$ to be the set of all finite lists with entries in $\mathcal{O}$ (we call these lists *stacks*), and $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ to be the set of all functions from $\mathrm{Stacks}(\mathcal{O})$ to itself. If $x, y \in \mathcal{O}$, then the *transporter set* from x to y in $\mathrm{Sym}(\Omega)$ is denoted by $\mathrm{Transp}(x, y) := \{g \in \mathrm{Sym}(\Omega) : x^g = y\}$, and the *stabiliser* of x in $\mathrm{Sym}(\Omega)$ is the subgroup $\mathrm{Stab}(x) := \mathrm{Transp}(x, x)$ of all permutations in $\mathrm{Sym}(\Omega)$ that fix x under the action. Note that for all $h \in \mathrm{Sym}(\Omega)$, $\mathrm{Transp}(x, y^h) = \mathrm{Transp}(x, y) \cdot h$.

The *setwise stabiliser* of a collection $x_1, \dots, x_n$ of objects is the stabiliser of the set $\{x_1, \dots, x_n\}$, whereas the *pointwise stabiliser* is $\bigcap_{i=1}^n \mathrm{Stab}(x_i)$, which is equal to the stabiliser of any list of the objects, such as $[x_1, \dots, x_n]$.

The backtrack search techniques considered in this article are organised around lists of points, ordered partitions, and graphs. An *ordered partition* of a set V is a list of nonempty disjoint subsets of V whose union is V. A *graph* on a set V is a pair (V, E) of *vertices* V and a set of 2-subsets of V called *edges*. Similarly, a *digraph* on V is a pair (V, A) of *vertices* V and a set of ordered pairs in V called *arcs*. The action of $\mathrm{Sym}(V)$ on the sets of all graphs and digraphs on V is defined, respectively, by $(V, E)^g := (V, E^g)$ and $(V, A)^g := (V, A^g)$ for all $g \in \mathrm{Sym}(V)$, edge sets E, and arc sets A. A *labelled digraph* is a digraph with an assignment of a label to each vertex and arc. See Jefferson et al. (2021, Section 2.1) for more information. Whenever, in this article, we write that '(V, E) is a graph', then this means that V is the set of vertices and E is the set of edges as explained above. In the same way, we use the remaining notation introduced here without further explanation.

2.1. Classical backtracking, partition backtracking, and graph backtracking

We briefly summarise the concept of backtrack search in $\mathrm{Sym}(\Omega)$, and introduce some terminology.

Let $U_1, \dots, U_k$ be subsets of $\mathrm{Sym}(\Omega)$ for some $k \in \mathbb{N}$, and suppose that we wish to search for the intersection $U_1 \cap \cdots \cap U_k$. For this technique to be useful in practice, it should be computationally cheap, for each $i \in \{1, \dots, k\}$, to determine whether any given element of $\mathrm{Sym}(\Omega)$ is contained in U_i.

Many typical search problems can be formulated in this way. For example, if U_1 is a subgroup of $\mathrm{Sym}(\Omega)$ given by generators and $U_2 := \mathrm{Transp}(\Gamma, \Delta)$ for some graphs Γ and Δ with vertex set Ω, then searching for an element of $U_1 \cap U_2$ solves the graph isomorphism problem for Γ and Δ in U_1.

Let $\mathcal{O}$ be a set on which $\mathrm{Sym}(\Omega)$ acts (such as Ω itself, or the set of all ordered partitions of Ω). In this paper we will present all search techniques in a common framework. The fundamental idea of this framework is to organise a backtrack search for $U_1 \cap \cdots \cap U_k$ around a pair of stacks of objects in $\mathcal{O}$. At any point in the algorithm, when the pair of stacks is (S, T) for some $S, T \in \mathrm{Stacks}(\mathcal{O})$, the set of permutations being searched is $\mathrm{Transp}(S, T)$. The stacks are initially empty, which means that the search begins with the whole symmetric group. In each step down into the recursion, new stacks are appended to the existing ones. This may be done by a refiner, in an attempt to remove redundant parts of the search space (Section 3), or by a splitter, in order to divide the search space into smaller parts that can be searched recursively (see Jefferson et al., 2021, Section 6).

For such a backtrack search to be practical, the set $\mathcal{O}$ should be easy to compute with, and in particular, it should be relatively cheap to compute stabilisers and transporter sets in $\mathrm{Sym}(\Omega)$ of elements in $\mathcal{O}$, or at least to obtain close overapproximations of them (see Jefferson et al., 2021,

Section 5). On the other hand, the set $\mathcal{O}$ should be sufficiently rich and varied that refiners can construct stacks in $\mathcal{O}$ that encode useful information about the problem at hand.

So far, backtrack search in finite symmetric groups has been formulated and implemented in several settings. We use the term *classical backtracking* for the case that $\mathcal{O} = \Omega$; this is essentially the original backtrack search in $\mathrm{Sym}(\Omega)$ introduced in Sims (1971), although presented differently. *Partition backtracking* is backtrack search where the objects are ordered partitions of Ω; this is essentially the technique introduced in Leon (1991). We use the term *graph backtracking* when the objects are labelled digraphs on Ω (Jefferson et al., 2021). In Section 6, we introduce an advancement of this latter technique, which we call *extended graph backtracking*.

3. Refiners for backtrack search

Throughout this section and Sections 4 and 5, we let $\mathcal{O}$ be a set on which $\mathrm{Sym}(\Omega)$ acts, so that the setting is backtrack search organised around stacks in $\mathcal{O}$. In particular, all definitions and results are relative to the set $\mathcal{O}$, even if we do not explicitly repeat that every single time.

Definition 3.1. A *refiner* for a set of permutations $U \subseteq \mathrm{Sym}(\Omega)$ is a pair (f_L, f_R) of functions in $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ such that:

$$\text{For all } S, T \in \mathrm{Stacks}(\mathcal{O}),\ U \cap \mathrm{Transp}(S,\ T) \subseteq \mathrm{Transp}(f_L(S),\ f_R(T)). \qquad (*)$$

In practice, we focus on refiners for subsets of $\mathrm{Sym}(\Omega)$ that are subgroups, cosets of subgroups, or empty.

We remark that any pair of functions in $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ is a refiner for the empty set and that it is necessary to include the empty set as a possibility because, for example, it is common to search for transporter sets, which may be empty.

Let (f_L, f_R) be a refiner for a set $U \subseteq \mathrm{Sym}(\Omega)$, let $S, T \in \mathrm{Stacks}(\mathcal{O})$ with $|S| = |T|$, and suppose that we are part way through a search for an intersection of subsets of $\mathrm{Sym}(\Omega)$ that includes U, with $\mathrm{Transp}(S, T)$ as the current search space. In this situation, the aim of applying this refiner to the stacks S and T is to prune elements of $\mathrm{Sym}(\Omega) \setminus U$ from the current search space, by moving to the transporter set corresponding to the lengthened stacks $S \parallel f_L(S)$ and $T \parallel f_R(T)$. By $(*)$, $\mathrm{Transp}(S \parallel f_L(S),\ T \parallel f_R(T))$ retains the elements of $\mathrm{Transp}(S, T)$ that *are* in U, but on the other hand, $\mathrm{Transp}(S \parallel f_L(S),\ T \parallel f_R(T))$ is contained in $\mathrm{Transp}(S, T)$, perhaps properly, and may therefore lack some elements of $\mathrm{Transp}(S, T)$ that are *not* in U. In particular, if $\mathrm{Transp}(S \parallel f_L(S),\ T \parallel f_R(T))$ is empty, then the search can backtrack.

Leon used the term *$\mathcal{P}$-refinement* in his work on partition backtracking (Leon, 1991, 1997) for a similar concept that is essentially compatible with our notion of a refiner, although he presents it in a significantly different fashion. Definition 3.1 matches the notion of a refiner used in Jefferson et al. (2021, Section 5).

We remark that the definition of a refiner guarantees no particular success at pruning. For example, if ι is the identity map on $\mathrm{Stacks}(\mathcal{O})$, and if ε is the constant map on $\mathrm{Stacks}(\mathcal{O})$ whose image is the empty stack in $\mathrm{Stacks}(\mathcal{O})$, then (ι, ι) and $(\varepsilon, \varepsilon)$ are refiners for every subset of $\mathrm{Sym}(\Omega)$, even though neither performs any pruning. Thus it is desirable to have a measure of the quality of a refiner. In the following section we introduce perfect refiners, which are those that have maximal pruning power.

To simplify some forthcoming exposition, we introduce a way of applying permutations to functions in $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$. It is straightforward to verify that this defines an action of $\mathrm{Sym}(\Omega)$.

Notation 3.2. For any $f \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ and $x \in \mathrm{Sym}(\Omega)$, we define f^x as follows:
$f^x(S) \coloneqq f(S^{x^{-1}})^x$ for all $S \in \mathrm{Stacks}(\mathcal{O})$.

Lemma 3.3. *For all $f \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ and $x \in \mathrm{Sym}(\Omega)$, the map f^x explained above is in $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$. Moreover, $\mathrm{Sym}(\Omega)$ acts on $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$.*

The following results show the close relationships between the two functions that comprise a refiner for a subgroup (Lemma 3.4) or more generally for a coset of a subgroup (Lemma 3.5). We note that Lemma 3.5 is a reformulation of Jefferson et al. (2021, Lemma 4.6) that uses Notation 3.2.

Lemma 3.4 *(Lemma 4.4 in Jefferson et al., 2021; cf. Leon, 1991, Lemma 6 and Leon, 1997, Prop 2). If* (f_L, f_R) *is a refiner for a subgroup of* $\mathrm{Sym}(\Omega)$*, then* $f_L = f_R$*.*

Proof. Suppose that (f_L, f_R) is a refiner for a group G and let $S \in \mathrm{Stacks}(\mathcal{O})$. Since $1_G \in \mathrm{Transp}(S, S)$, it follows by $(*)$ that $1_G \in \mathrm{Transp}(f_L(S), f_R(S))$, which means that $f_L(S) = f_L(S)^{1_G} = f_R(S)$. □

Lemma 3.5. *Let* $G \leq \mathrm{Sym}(\Omega)$*,* $x \in \mathrm{Sym}(\Omega)$*, and* $f_L, f_R \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$*. Then* (f_L, f_R) *is a refiner for the right coset* Gx *if and only if* (f_L, f_L) *is a refiner for* G *and* $f_R = {f_L}^x$*.*

By Lemmas 3.4 and 3.5, a refiner for a group or coset is built from a single function from $\mathrm{Stacks}(\mathcal{O})$ to itself. The following lemma gives equivalent conditions for such a function to yield a refiner (always with respect to the set $\mathcal{O}$, as mentioned earlier):

Lemma 3.6. *Let* $G \leq \mathrm{Sym}(\Omega)$ *and* $f \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$*. The following are equivalent:*

(i) (f, f) *is a refiner for* G*.*
(ii) $f(S^x) = f(S)^x$ *for all* $S \in \mathrm{Stacks}(\mathcal{O})$ *and* $x \in G$*.*
(iii) $G \leq \mathrm{Stab}(f)$*.*

Proof. Suppose that (i) holds and let $S \in \mathrm{Stacks}(\mathcal{O})$ and $x \in G$. By Definition 3.1, $G \cap \mathrm{Transp}(S, S^x) \subseteq \mathrm{Transp}(f(S), f(S^x))$. Then (ii) holds, since $x \in G \cap \mathrm{Transp}(S, S^x)$. Conversely, suppose that (ii) holds and let $S, T \in \mathrm{Stacks}(\mathcal{O})$. If $x \in G \cap \mathrm{Transp}(S, T)$, then $T = S^x$, and since $f(S)^x = f(S^x)$ by assumption, it follows that $x \in \mathrm{Transp}(f(S), f(T))$. Therefore (i) holds by Definition 3.1.

The equivalence of (ii) and (iii) is clear from Notation 3.2 and the definition of $\mathrm{Stab}(f)$. □

Next, we show a way for refiners to be combined to give a refiner for an intersection of sets.

Notation 3.7. For all $f, g \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$, we define the function $f \parallel g \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ by $(f \parallel g)(S) := f(S) \parallel g(S)$ for all $S \in \mathrm{Stacks}(\mathcal{O})$.

Lemma 3.8. *Let* (f, σ) *be a refiner for the set* $U \subseteq \mathrm{Sym}(\Omega)$*, and let* (g, τ) *be a refiner for* $V \subseteq \mathrm{Sym}(\Omega)$*. Then* $(f \parallel g, \sigma \parallel \tau)$ *is a refiner for* $U \cap V$*.*

Proof. We show that the set $U \cap V$ and the pair of functions $(f \parallel g, \sigma \parallel \tau)$ satisfy $(*)$. Let $S, T \in \mathrm{Stacks}(\mathcal{O})$. If $|f(S)| \neq |\sigma(T)|$, then $\mathrm{Transp}(f(S), \sigma(T)) = \varnothing$, and so $U \cap \mathrm{Transp}(S, T) = \varnothing$, since (f, σ) is a refiner for U. In particular, $(U \cap V) \cap \mathrm{Transp}(S, T) = \varnothing$. If instead $|f(S)| = |\sigma(T)|$, then

$$\begin{aligned}(U \cap V) \cap \mathrm{Transp}(S, T) &= (U \cap \mathrm{Transp}(S, T)) \cap (V \cap \mathrm{Transp}(S, T)) \\ &\subseteq \mathrm{Transp}(f(S), \sigma(T))) \cap \mathrm{Transp}(g(S), \tau(T)) \\ &= \mathrm{Transp}(f(S) \| g(S), \sigma(T) \parallel \tau(T)) \\ &= \mathrm{Transp}((f \parallel g)(S), (\sigma \parallel \tau)(T)).\end{aligned}$$ □

We remark that the $\parallel$ operation is associative, and that we may repeatedly apply Lemma 3.8 to obtain a refiner for any finite intersection of sets, given a refiner for each set.

4. Perfect refiners

Lemma 4.1. *Let $U \subseteq \mathrm{Sym}(\Omega)$, let $f_L, f_R \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$, and suppose that*

$$U \cap \mathrm{Transp}(S, T) = \mathrm{Transp}(S \parallel f_L(S), T \parallel f_R(T)) \qquad (\circledast)$$

for all $S, T \in \mathrm{Stacks}(\mathcal{O})$ with $|S| = |T|$. Then (f_L, f_R) is a refiner for U.

Proof. To show that $(*)$ holds, let $S, T \in \mathrm{Stacks}(\mathcal{O})$. If $\mathrm{Transp}(S, T) = \varnothing$, then this is clear, so suppose otherwise; in particular, suppose that $|S| = |T|$. Then

$$\begin{aligned} U \cap \mathrm{Transp}(S, T) &= \mathrm{Transp}(S \parallel f_L(S), T \parallel f_R(T)) \\ &= \mathrm{Transp}(S, T) \cap \mathrm{Transp}(f_L(S), f_R(T)) \\ &\subseteq \mathrm{Transp}(f_L(S), f_R(T)). \quad \square \end{aligned}$$

Definition 4.2. Refiners with the property $(\circledast)$ from Lemma 4.1 are called *perfect refiners* (with respect to $\mathcal{O}$, which we will usually omit).

We give several examples of perfect refiners in Section 8. Note that a refiner for a set $U \subseteq \mathrm{Sym}(\Omega)$ is also a refiner for any proper subset of U, but never a perfect refiner. The equation '$|S| = |T|$' in $(\circledast)$ is included for convenience: in principle, two stacks of different lengths (whose transporter set is empty) could be extended to stacks with nonempty transporter set. However, a refiner is only ever applied to pairs of stacks of equal lengths, so we remove the need to deal with this complication.

During a search, a perfect refiner for U can be used to extend the stacks S and T (representing the current search space $\mathrm{Transp}(S, T)$) to obtain the potentially-smaller search space $\mathrm{Transp}(S, T) \cap U$. Thus a perfect refiner for U allows U to be represented with full fidelity, and no effort must be wasted considering elements of $\mathrm{Sym}(\Omega)$ that are not in U. In other words: Perfect refiners are the refiners with maximal pruning power.

A perfect refiner therefore needs to be applied at most once in any branch of search, and should therefore be applied at the root node to avoid unnecessary repetition. It is for this reason that perfect refiners often comprise constant functions in practice; see Lemma 4.3.

Note that if each set U_i in a search for $U_1 \cap \cdots \cap U_n$ is given with a perfect refiner, then the search can terminate at the root node without splitting or backtracking. This is because the search begins with S and T being empty stacks, and so their transporter set is initially $\mathrm{Sym}(\Omega)$. Applying the perfect refiners in turn then gives stacks with transporter set $U_1 \cap \cdots \cap U_n$, as required.

In the following lemmas, we see that constructing a perfect refiner for a set U is equivalent to finding a pair of stacks in $\mathcal{O}$ with transporter set U. It follows that the existence of a perfect refiner for any given subset of $\mathrm{Sym}(\Omega)$, in backtrack search organised around stacks in $\mathcal{O}$, depends on the choice of $\mathcal{O}$. But as before, we will not always add "with respect to $\mathcal{O}$".

Lemma 4.3. *Let $U \subseteq \mathrm{Sym}(\Omega)$, and suppose there exist $A, B \in \mathrm{Stacks}(\mathcal{O})$ such that $U \subseteq \mathrm{Transp}(A, B)$. Let f_A and f_B be constant functions on $\mathrm{Stacks}(\mathcal{O})$ with images A and B, respectively. Then (f_A, f_B) is a refiner for U. Moreover, if $U = \mathrm{Transp}(A, B)$, then this refiner is perfect.*

Proof. The pair of functions satisfy the condition in Definition 3.1, since for all $S, T \in \mathrm{Stacks}(\mathcal{O})$:

$$\mathrm{Transp}(S, T) \cap U \subseteq U \subseteq \mathrm{Transp}(A, B) = \mathrm{Transp}(f_A(S), f_B(T)).$$

If $U = \mathrm{Transp}(A, B)$, then for all $S, T \in \mathrm{Stacks}(\mathcal{O})$ with $|S| = |T|$, condition $(\circledast)$ is satisfied, since

$$\begin{aligned} \mathrm{Transp}(S, T) \cap U &= \mathrm{Transp}(S, T) \cap \mathrm{Transp}(A, B) \\ &= \mathrm{Transp}(S, T) \cap \mathrm{Transp}(f_A(S), f_B(T)) \\ &= \mathrm{Transp}(S \parallel f_A(S), T \parallel f_B(T)). \quad \square \end{aligned}$$

Corollary 4.4. *Let $G \leq \mathrm{Sym}(\Omega)$, and suppose there exists $A \in \mathrm{Stacks}(\mathcal{O})$ such that $G \leq \mathrm{Stab}(A)$. Define f_A to be the constant function on $\mathrm{Stacks}(\mathcal{O})$ with image A. Then (f_A, f_A) is a refiner for G. Moreover, if $G = \mathrm{Stab}(A)$, then this refiner is perfect.*

Lemma 4.5. *Let $U \subseteq \mathrm{Sym}(\Omega)$. There exists a perfect refiner for U if and only if $U = \mathrm{Transp}(S, T)$ for some $S, T \in \mathrm{Stacks}(\mathcal{O})$. In particular, there exists a perfect refiner for a group $G \leq \mathrm{Sym}(\Omega)$ if and only if $G = \mathrm{Stab}(S)$ for some $S \in \mathrm{Stacks}(\mathcal{O})$.*

Proof. Let (f_L, f_R) be a perfect refiner for U and let S be the empty stack in $\mathrm{Stacks}(\mathcal{O})$. Note that $\mathrm{Transp}(S, S) = \mathrm{Sym}(\Omega)$. Then $U = \mathrm{Transp}(f_L(S), f_R(S))$ by Definition 4.2. The converse implication follows from Lemma 4.3. □

Corollary 4.6. *If there exists a perfect refiner for a subset $U \subseteq \mathrm{Sym}(\Omega)$, then either U is empty, or U is a subgroup, or a coset of a subgroup, of $\mathrm{Sym}(\Omega)$.*

Lemma 4.7 *(cf. Lemma 3.5). Let $G \leq \mathrm{Sym}(\Omega)$, let $x \in \mathrm{Sym}(\Omega)$, and let $f_L, f_R \in \mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$. Then (f_L, f_R) is a perfect refiner for Gx if and only if (f_L, f_L) is a perfect refiner for G and $f_R = f_L^x$. In particular, there exists a perfect refiner for a group $G \leq \mathrm{Sym}(\Omega)$ if and only if there exist perfect refiners for one, and hence all, cosets of G in $\mathrm{Sym}(\Omega)$.*

Proof. We prove that for all $x, y \in \mathrm{Sym}(\Omega)$, (f_L, f_L^x) is a perfect refiner for Gx if and only if (f_L, f_L^y) is a perfect refiner for Gy. The lemma follows from this by choosing $y := 1_G$ and by Lemma 3.5.

Let $S, T \in \mathrm{Stacks}(\mathcal{O})$ with $|S| = |T|$. Suppose that (f_L, f_L^x) is a perfect refiner for Gx. Note that $Gy \cap \mathrm{Transp}(S, T) = (Gx \cap \mathrm{Transp}(S, T^{y^{-1}x})) \cdot x^{-1}y$. Then by assumption:

$$\begin{aligned}
Gx \cap \mathrm{Transp}(S, T^{y^{-1}x}) &= \mathrm{Transp}(S \parallel f_L(S), T^{y^{-1}x} \parallel f_L^x(T^{y^{-1}x})) \\
&= \mathrm{Transp}(S \parallel f_L(S), T^{y^{-1}x} \parallel f_L^y(T)^{y^{-1}x}) \\
&= \mathrm{Transp}(S \parallel f_L(S), (T \parallel f_L^y(T))^{y^{-1}x}) \\
&= \mathrm{Transp}(S \parallel f_L(S), T \parallel f_L^y(T)) \cdot y^{-1}x.
\end{aligned}$$

Hence (f_L, f_L^y) is a perfect refiner for Gy by (⊛). The converse implication follows by symmetry. □

The following lemma shows that combining perfect refiners with the $\parallel$ operation of Notation 3.7 preserves perfectness. This implies that the collection of groups and cosets which have perfect refiners is closed under intersection.

Lemma 4.8. *Let (f, σ) and (g, τ) be perfect refiners for the sets $U, V \subseteq \mathrm{Sym}(\Omega)$, respectively. Then $(f \| g, \sigma \| \tau)$ is a perfect refiner for the set $U \cap V$.*

Proof. We show that condition (⊛) from Lemma 4.1 holds, for the set $U \cap V$ and the pair of functions $(f \| g, \sigma \| \tau)$. Let $S, T \in \mathrm{Stacks}(\mathcal{O})$ with $|S| = |T|$. By an argument similar to that used in the proof of Lemma 3.8, if $|f(S)| \neq |\sigma(T)|$, then the condition holds, so suppose otherwise. Then

$$\begin{aligned}
(U \cap V) \cap \mathrm{Transp}(S, T) &= (U \cap \mathrm{Transp}(S, T)) \cap (V \cap \mathrm{Transp}(S, T)) \\
&= \mathrm{Transp}(S \| f(S), T \| \sigma(T)) \cap \mathrm{Transp}(S \| g(S), T \| \tau(T)) \\
&= \mathrm{Transp}(S, T) \cap \mathrm{Transp}(f(S), \sigma(T)) \cap \mathrm{Transp}(g(S), \tau(T)) \\
&= \mathrm{Transp}(S \cap (f \| g)(S), T \cap (\sigma \| \tau)(T)). \quad \square
\end{aligned}$$

5. Perfect refiners for stabilisers and transporter sets

Many computations that are commonly performed with backtrack search can be formulated as stabiliser or transporter problems. This includes computing normalisers, determining subgroup conjugacy, solving graph isomorphism, and finding sets that are phrased in such terms, like set stabilisers.

Therefore, in order to most successfully solve such problems with backtrack search organised around stacks in $\mathcal{O}$, we require a way of constructing refiners for the appropriate stabilisers and transporter sets. These refiners should be cheap to compute, and ideally, they should be perfect if possible.

Lemma 4.3 constructively shows that this task can be reduced to translating the given stabiliser or transporter problem into one concerning stacks in $\mathcal{O}$. For a problem that is already given in terms of Stacks($\mathcal{O}$), or at least in terms of $\mathcal{O}$, it follows that little to no translation is needed; this is stated explicitly in the following immediate corollary to Lemma 4.3.

Corollary 5.1. *For each $x \in \mathcal{O}$ and $S \in \mathrm{Stacks}(\mathcal{O})$, let $[x]$ be the stack with unique entry x, and let f_S be the constant function in $\mathrm{Stacks}(\mathcal{O})^{\mathrm{Stacks}(\mathcal{O})}$ with image S. Let $x, y \in \mathcal{O}$ and $S, T \in \mathrm{Stacks}(\mathcal{O})$. Then $(f_{[x]}, f_{[y]})$ is a perfect refiner for $\mathrm{Transp}(x, y)$, and (f_S, f_T) is a perfect refiner for $\mathrm{Transp}(S, T)$.*

For other transporter problems, more work is required in order to apply Lemma 4.3, i.e. to construct stacks in $\mathcal{O}$ whose own transporter set closely contains (and ideally equals) the given one. The analogous statement holds for stabiliser problems. While this translation can be done on an ad-hoc basis, it is desirable to instead have a systematic method that works for a whole class of objects that we wish to stabilise and to compute transporter sets of. To that end, we present Theorem 5.3.

Definition 5.2. Let X and Y be sets on which $\mathrm{Sym}(\Omega)$ acts, and let $f : X \to Y$ be a function. Then f is called $\mathrm{Sym}(\Omega)$-*invariant* if $f(x^g) = f(x)^g$ for all $x \in X$ and $g \in G$.

Theorem 5.3. *Let Σ and Θ be sets on which $\mathrm{Sym}(\Omega)$ acts, let $\pi : \Sigma \to \Theta$ be a $\mathrm{Sym}(\Omega)$-invariant function, and let $x, y \in \Sigma$. Then the following statements hold:*

(i) $\mathrm{Transp}(x, y) \subseteq \mathrm{Transp}(\pi(x), \pi(y))$. *In particular,* $\mathrm{Stab}(x) \leq \mathrm{Stab}(\pi(x))$, *and any refiner for* $\mathrm{Transp}(\pi(x), \pi(y))$ *is a refiner for* $\mathrm{Transp}(x, y)$.

(ii) *If π is injective, then* $\mathrm{Transp}(x, y) = \mathrm{Transp}(\pi(x), \pi(y))$*; in particular,* $\mathrm{Stab}(x) = \mathrm{Stab}(\pi(x))$, *and any perfect refiner for* $\mathrm{Transp}(\pi(x), \pi(y))$ *is a perfect refiner for* $\mathrm{Transp}(x, y)$.

Proof. If $g \in \mathrm{Transp}(x, y)$, then $\pi(x)^g = \pi(x^g) = \pi(y)$, i.e. $g \in \mathrm{Transp}(\pi(x), \pi(y))$. Therefore (i) holds. Suppose that π is injective. If $g \in \mathrm{Transp}(\pi(x), \pi(y))$, then $\pi(x)^g = \pi(x^g)$ and $\pi(x)^g = \pi(y)$, and so the injectivity of π implies that $x^g = y$. Therefore (ii) holds. □

Theorem 5.3 suggests a strategy for systematically constructing refiners for the stabilisers and transporter sets of objects in a set on which $\mathrm{Sym}(\Omega)$ acts, which is the foundation of our approach in Section 8. More precisely, given such a set Σ, we identify another set Θ on which $\mathrm{Sym}(\Omega)$ acts, and for which we have identified refiners for all stabilisers and transporter sets. We then define an injective $\mathrm{Sym}(\Omega)$-invariant function $\pi : \Sigma \to \Theta$. The desired refiners are then inherited via π as described in Theorem 5.3, with Corollary 5.1 providing the base of this recursive procedure.

Remark 5.4. Let $\mathcal{O}$ and $\mathcal{O}'$ be sets on which $\mathrm{Sym}(\Omega)$ acts, let $\pi : \mathrm{Stacks}(\mathcal{O}) \to \mathrm{Stacks}(\mathcal{O}')$ be an injective $\mathrm{Sym}(\Omega)$-invariant function, and let $U \subseteq \mathrm{Sym}(\Omega)$. Then Lemmas 4.3 and 4.5 and Theorem 5.3(ii) together constructively show that if U has a perfect refiner in backtrack search organised around Stacks($\mathcal{O}$), then U also has a perfect refiner in backtrack search organised around Stacks($\mathcal{O}'$).

In Section 8, we also make frequent use of the following lemma, which shows that the $\|$ operation can be used to construct refiners for the stabilisers and transporter sets of lists. This means that lists do need not to be considered separately from the objects that they contain.

Lemma 5.5. *Let $n \in \mathbb{N}$ with $n \geq 2$. For each $i \in \{1, \ldots, n\}$, let x_i and y_i be objects from a set on which $\mathrm{Sym}(\Omega)$ acts, and let $(f_{L,i}, f_{R,i})$ be a refiner for $\mathrm{Transp}(x_i, y_i)$. Then $(f_{L,1} \| \cdots \| f_{L,n}, f_{R,1} \| \cdots \| f_{R,n})$ is a refiner for $\mathrm{Transp}([x_1, \ldots, x_n], [y_1, \ldots, y_n])$. Moreover, if each refiner $(f_{L,i}, f_{R,i})$ is perfect, then the resulting refiner is perfect, too.*

Proof. Since $\mathrm{Transp}([x_1, \ldots, x_n], [y_1, \ldots, y_n]) = \bigcap_{i=1}^{n} \mathrm{Transp}(x_i, y_i)$, the result follows from Lemmas 3.8 and 4.8. □

Corollary 5.6. *Let Σ be a set on which $\mathrm{Sym}(\Omega)$ acts. Then there exist perfect refiners for all stabilisers and for all transporter sets of objects in Σ if and only if there exist perfect refiners for all stabilisers and transporter sets of finite lists in Σ.*

6. Extended graph backtracking, with extra vertices

With graph backtracking, a search in $\mathrm{Sym}(\Omega)$ is organised around stacks of labelled digraphs on the vertex set Ω. Although this technique gives significantly smaller search sizes in some cases (Jefferson et al., 2021, Section 9), and enables some important perfect refiners (Section 8.2), the requirement that the digraphs have vertex set Ω limits the possibilities for perfect refiners. This is shown explicitly in Corollary 7.3.

In this section, we introduce an extension to graph backtracking, which accommodates digraphs that are allowed to have additional vertices. The technique requires only a small adjustment to the theory of graph backtracking, and has already been implemented in Vole (Chang et al., 2021). In fact the necessary concepts from Jefferson et al. (2021) extend naturally, and re-stating and re-proving the corresponding definitions and results do not give any new insights, which is why we do not include these technical details in this article. Instead, we discuss the extended graphs themselves, and their stacks, in much detail, because this is necessary for formulating and examining refiners in this setting.

We require some additional definitions and notation for the rest of this article.

We fix Λ as an infinite well-ordered set containing Ω, where $\alpha < \beta$ for all $\alpha \in \Omega$ and $\beta \in \Lambda \setminus \Omega$. In practice, and in the forthcoming examples, we use $\Omega := \{1, \ldots, n\}$ for some $n \in \mathbb{N}$, and $\Lambda := \mathbb{N}$.

For any subset V of Λ that contains Ω, we regard $\mathrm{Sym}(\Omega)$ and $\mathrm{Sym}(V \setminus \Omega)$ as subgroups of $\mathrm{Sym}(V)$, by identifying each permutation of V that fixes $V \setminus \Omega$ pointwise with its restriction to Ω, and identifying each permutation of V that fixes Ω pointwise with its restriction to $V \setminus \Omega$. In this way, $\mathrm{Sym}(\Omega)$ and $\mathrm{Sym}(V \setminus \Omega)$ inherit from $\mathrm{Sym}(\Omega)$ an action on the set of all labelled digraphs on V. Furthermore, since we regard $\mathrm{Sym}(\Omega)$ and $\mathrm{Sym}(V \setminus \Omega)$ as subgroups of $\mathrm{Sym}(V)$, they commute, and this means that $\mathrm{Sym}(\Omega)$ permutes the set of orbits of $\mathrm{Sym}(V \setminus \Omega)$ on labelled digraphs on V. For a labelled digraph Γ on V, we use the notation $\overline{\Gamma}$ for the orbit of Γ under $\mathrm{Sym}(V \setminus \Omega)$. The action of $\mathrm{Sym}(\Omega)$ is then given by $(\overline{\Gamma})^g := \overline{\Gamma^g}$ for all labelled digraphs Γ on V and all $g \in \mathrm{Sym}(\Omega)$.

We define an *extended graph* on Ω to be an orbit of $\mathrm{Sym}(V \setminus \Omega)$ on the set of labelled digraphs on V, for some finite set V with $\Omega \subseteq V \subseteq \Lambda$. This will be illustrated in Example 6.1. An *extended graph stack* on Ω is any list of extended graphs on Ω. We remark that different entries in an extended graph stack are allowed to be defined in relation to different subsets of Λ. *Extended graph backtracking* in $\mathrm{Sym}(\Omega)$ is then backtrack search organised around extended graph stacks on Ω.

The results of Sections 3–5 hold in the setting of extended graph backtracking in $\mathrm{Sym}(\Omega)$.

Example 6.1. Let $\Omega := \{1, \ldots, 4\}$ and $V := \{1, \ldots, 6\}$. Depicted in Fig. 1 are labelled digraphs Γ and $\Gamma^{(5\,6)}$ on V, which comprise an orbit of $\mathrm{Sym}(V \setminus \Omega)$ on the set of labelled digraphs on V. Therefore $\overline{\Gamma} = \{\Gamma, \Gamma^{(5\,6)}\}$ is an extended graph on Ω. Note that $\mathrm{Stab}(\overline{\Gamma}) = \langle (1\,4), (1\,2)(3\,4) \rangle$, which is the setwise stabiliser of $\{\{1, 4\}, \{2, 3\}\}$ in $\mathrm{Sym}(\Omega)$. In Section 8.3.1 we will discuss a similar construction.

Lemma 6.2. *Let A and B be extended graphs on Ω, defined relative to subsets U and V of Λ, respectively, and let $\Gamma \in A$ and $\Delta \in B$ be arbitrary. If $U \neq V$, then $\mathrm{Transp}(U, V) = \varnothing$. Otherwise, $\mathrm{Transp}(A, B)$ is the restriction of $T := \{x \in \mathrm{Sym}(V) : x$ preserves Ω setwise, and $\Gamma^x = \Delta\}$ to Ω.*

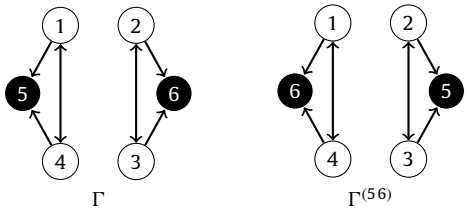

Fig. 1. The digraphs Γ and $\Gamma^{(56)}$ from Example 6.1.

Table 1
This table shows the subgroups of $\mathrm{Sym}(\Omega)$, and the cosets of subgroups of $\mathrm{Sym}(\Omega)$, for which there exists a perfect refiner, in various kinds of backtracking. For a subset $\Sigma \subseteq \Omega$, we identify $\mathrm{Sym}(\Sigma)$ with the pointwise stabiliser of $\Omega \setminus \Sigma$ in $\mathrm{Sym}(\Omega)$, and for a partition $\{\Sigma_1, \ldots, \Sigma_k\}$ of Ω, we identify $\mathrm{Sym}(\Sigma_1) \times \cdots \times \mathrm{Sym}(\Sigma_k)$ with the corresponding internal direct product in $\mathrm{Sym}(\Omega)$.

Backtracking	The subgroups of $\mathrm{Sym}(\Omega)$, and their cosets, with perfect refiners	Reference
Classical	$\mathrm{Sym}(\Sigma)$ for any subset $\Sigma \subseteq \Omega$.	–
Partition	$\mathrm{Sym}(\Sigma_1) \times \cdots \times \mathrm{Sym}(\Sigma_k)$ for any partition $\{\Sigma_1, \ldots, \Sigma_k\}$ of Ω.	–
Graph	Any 2-closed subgroup of $\mathrm{Sym}(\Omega)$.	Corollary 7.3
Extended graph	Any subgroup of $\mathrm{Sym}(\Omega)$.	Corollary 7.5

Proof. Suppose that $U = V$ and let g be the restriction to Ω of some $x \in T$. Then since $g^{-1}x \in \mathrm{Sym}(V \setminus \Omega)$, it follows that $A^g = \overline{\Gamma}^g = \overline{\Gamma^g} = \overline{\Gamma^{g(g^{-1}x)}} = \overline{\Gamma^x} = \overline{\Delta} = B$. Conversely, if $g \in \mathrm{Transp}(A, B) = \mathrm{Transp}(\overline{\Gamma}, \overline{\Delta})$, then $\Gamma^g = \Delta^h$ for some $h \in \mathrm{Sym}(V \setminus \Omega)$. Thus g is the restriction of $gh^{-1} \in T$ to Ω. □

Lemma 6.2 implies that we do not need to enumerate whole orbits of labelled digraphs in order to compute stabilisers and transporter sets of extended graphs. In practice, we construct extended graphs that consist of labelled digraphs where the labels used for vertices in Ω are never used for vertices in $\Lambda \setminus \Omega$. This guarantees that any permutation that maps one such labelled digraph to another necessarily preserves Ω as a set. Therefore, in the notation of Lemma 6.2, the set T is simply the transporter set in $\mathrm{Sym}(V)$ from Γ to Δ.

7. The groups and cosets with perfect refiners

By Lemmas 4.5 and 4.7, the task of finding a perfect refiner for a subgroup of $\mathrm{Sym}(\Omega)$, or for a coset of a subgroup, is equivalent to finding a stack whose stabiliser in $\mathrm{Sym}(\Omega)$ is the corresponding group. In this section, and for each setting of backtracking, we classify the groups (and correspondingly the cosets of subgroups) for which there exists a perfect refiner. This information is summarised in Table 1.

It follows from these classifications that, in a sense, partition backtracking inherits the perfect refiners of classical backtracking, that graph backtracking inherits the perfect refiners of partition backtracking, and that extended graph backtracking inherits the perfect refiners of graph backtracking. This can be shown constructively: it is straightforward to map lists of points to lists of ordered partitions, and lists of ordered partitions to lists of labelled digraphs, in injective $\mathrm{Sym}(\Omega)$-invariant ways. As mentioned in Remark 5.4, perfect refiners can then be translated via these maps.

The classifications in Table 1 for classical backtracking (stabilisers in $\mathrm{Sym}(\Omega)$ of lists in Ω) and partition backtracking (stabilisers in $\mathrm{Sym}(\Omega)$ of lists of ordered partitions of Ω) are trivial to verify.

For graph backtracking, we require the notion of the 2-closure of a permutation group.

Definition 7.1 *(2-closure, 2-closed; cf. Dixon and Mortimer, 1996, Section 3.2).* The *2-closure* of a group $G \leq \mathrm{Sym}(\Omega)$ is the largest subgroup of $\mathrm{Sym}(\Omega)$ with the same orbits on $\Omega \times \Omega$ as G; it is the pointwise stabiliser in G of the set of all orbital graphs of G. A *2-closed* group is one that is equal to its 2-closure.

The 2-closure of any subgroup of $\mathrm{Sym}(\Omega)$ that acts at least 2-transitively on Ω is $\mathrm{Sym}(\Omega)$. Therefore no proper subgroup of $\mathrm{Sym}(\Omega)$ acts at least 2-transitively on Ω and is 2-closed.

Lemma 7.2. *Let G be a subgroup of $\mathrm{Sym}(\Omega)$. Then G is the stabiliser of a labelled digraph stack on Ω if and only if G is 2-closed.*

Proof. ($\Leftarrow$) Labelling all vertices and arcs of a digraph with a fixed label preserves its stabiliser. Thus, if S is a stack of labelled digraphs formed from all the orbital graphs of G in this way, then $G = \mathrm{Stab}(S)$.

($\Rightarrow$) By Jefferson et al. (2021, Lemma 3.6), we may assume that $G = \mathrm{Stab}(\Gamma)$ for a labelled digraph Γ on Ω. Let $A_1, \dots, A_k \subseteq \Omega \times \Omega$ and $B_1, \dots, B_l \subseteq \Omega$ be the orbits of G on the sets of arcs and vertices of Γ, respectively, for some $k \in \mathbb{N} \cup \{0\}$ and $l \in \mathbb{N}$. For each $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, l\}$, we define digraphs $\Delta_i := (\Omega, A_i)$ and $\Sigma_j := (\Omega, \{(b, b) : b \in B_j\})$. Each such digraph is an orbital graph of G, and so its stabiliser contains G. Furthermore, if a permutation $g \in \mathrm{Sym}(\Omega)$ stabilises each of these digraphs pointwise, then by construction, g maps each vertex of Γ to a vertex with the same label, and it maps each arc of Γ to an arc with the same label, and so $g \in \mathrm{Stab}(\Gamma)$. Thus we have proved that

$$G \leq \left(\bigcap_{i=1}^{k} \mathrm{Stab}(\Delta_i) \right) \cap \left(\bigcap_{j=1}^{l} \mathrm{Stab}(\Sigma_j) \right) \leq \mathrm{Stab}(\Gamma) = G.$$

Since G is the pointwise stabiliser of a subset of its orbital graphs, it is 2-closed. □

Corollary 7.3. *In graph backtracking, there exists a perfect refiner for a group $G \leq \mathrm{Sym}(\Omega)$ if and only if G is 2-closed.*

For extended graph backtracking, we require the following result.

Proposition 7.4 *(Bouwer, 1969; Kearnes, 2015). Let $G \leq \mathrm{Sym}(\Omega)$. There exists a finite set V containing Ω and a labelled digraph Γ on V, such that the stabiliser of Γ in $\mathrm{Sym}(V)$ preserves Ω setwise, and the restriction of this stabiliser to Ω is G.*

Lemma 7.5. *Every subgroup of $\mathrm{Sym}(\Omega)$ has a perfect refiner in extended graph backtracking.*

Proof. Let $G \leq \mathrm{Sym}(\Omega)$, and let Γ be a finite labelled digraph of the kind described in Proposition 7.4 for G. By renaming the vertices that are not in Ω, we may assume that the vertex set of Γ is a subset of the infinite set Λ that we use throughout this article. By Lemma 6.2 and Proposition 7.4, the extended graph stack $[\overline{\Gamma}]$ has stabiliser G in $\mathrm{Sym}(\Omega)$. The result follows by Lemma 4.5. □

We primarily include Lemma 7.5 for its theoretical interest. Given a group G, the number of additional vertices required to construct a witness for the result in Proposition 7.4 may be of similar size to $|G|$, which would normally be impractically large for use in a backtrack search. We are therefore also interested in how many extra vertices are required to represent different groups.

8. Examples of perfect refiners

In this section, we describe some refiners for the stabilisers and transporter sets of some commonly-occurring objects on which $\mathrm{Sym}(\Omega)$ acts. In most cases, we prove that the refiners are perfect, making frequent use of Corollary 5.1, Theorem 5.3, and Lemma 5.5. The results for perfect refiners are summarised in Table 2. As discussed in Section 7, a perfect refiner for a subset of $\mathrm{Sym}(\Omega)$ in classical backtracking can be easily converted into a perfect refiner for the same set in partition

Table 2
For the various kinds of backtracking in Sym(Ω), this table gives some objects on which Sym(Ω) acts, such that all stabilisers and transporter sets in Sym(Ω) of these objects have perfect refiners. A constructive argument is given in the reference. This table is not meant to be exhaustive.

Backtracking	Stabilisers/transporter sets in Sym(Ω) with perfect refiners	Reference
Classical	Point in Ω.	Corollary 5.1
	List in Ω.	Corollary 5.1
Partition	Ordered partition of Ω.	Corollary 5.1
	List of ordered partitions of Ω.	Corollary 5.1
	Subset of Ω.	Section 8.1.1
	List of subsets of Ω.	Lemma 5.5
	Set of subsets of Ω with pairwise distinct sizes.	Section 8.1.2
Graph	Labelled digraph on Ω.	Corollary 5.1
	Graph or digraph on Ω.	Section 8.2.1
	Set of nonempty pairwise disjoint subsets of Ω.	Section 8.2.2
	Unordered partition of Ω.	Section 8.2.2
	Permutation under conjugation (elt. centraliser/conjugacy).	Section 8.2.3
	List of permutations under conj. (subset/subgroup centraliser).	Lemma 5.5
Extended graph	Set of subsets of Ω.	Section 8.3.1
	Set of lists in Ω.	Section 8.3.2
	Set of (labelled) graphs or digraphs on Ω.	Section 8.3.3
	Set of lists of (labelled) graphs or digraphs on Ω.	Section 8.3.4
	2-closed subgroup under conjugation (normaliser/conjugacy).	Section 8.3.5

backtracking, and so on. In this hierarchical sense, each class of objects in Table 2 appears by the 'least' kind of backtracking in which all of their stabilisers and transporter sets have perfect refiners.

A given kind of backtracking may not have perfect refiners for *all* stabilisers and transporter sets of some class of objects, but it may have perfect refiners for those of an important subclass of the objects. In order to optimise implementations, it is important to understand these special cases well, but a thorough investigation of this topic is beyond the scope of this article. Nevertheless, to demonstrate the principle, we consider sets of subsets of Ω with pairwise distinct sizes in partition backtracking (Section 8.1.2), and sets of nonempty pairwise disjoint subsets of Ω in graph backtracking (Section 8.2.2), even though only extended graph backtracking has perfect refiners for all stabilisers and transporter sets of sets of subsets of Ω (Section 8.3.1).

8.1. Examples of perfect refiners in partition backtracking

To justify the part of Table 2 that relates to partition backtracking, we first show, for each kind of object, and with $|\Omega| \geq 5$, that not all stabilisers and transporter sets have perfect refiners in classical backtracking. By Corollary 5.6 and Table 1, the following examples suffice: If $n \in \mathbb{N}$ with $n \geq 5$ and $\Omega := \{1, \ldots, n\}$, then the ordered partition $[\{1,2\}, \{3, \ldots, n\}]$ of Ω, the subset $\{1,2\}$ of Ω, and the set of subsets $\{\{1,2\}, \{3, \ldots, n\}\}$ of Ω have stabiliser $\langle (1\,2), (3\,4), (3\,\ldots\,n)\rangle \cong \mathcal{C}_2 \times \mathcal{S}_{n-2}$ in Sym(Ω).

8.1.1. Stabilisers and transporters of sets of points

We define a function π from the set of all subsets of Ω to the set of all stacks of ordered partitions of Ω. We define $\pi(\varnothing)$ to be the empty stack on Ω, and $\pi(\Omega)$ to be the stack consisting of the ordered partition $[\Omega]$, and for any nonempty proper subset A of Ω, we define $\pi(A)$ to be the stack consisting of the ordered partition $[A, \Omega \setminus A]$. Since π is an injective Sym(Ω)-invariant function, Theorem 5.3 and Corollary 5.1 give the desired perfect refiners.

8.1.2. Stabilisers and transporters of sets of subsets with pairwise distinct sizes

We define an injective Sym(Ω)-invariant function σ that maps each set of subsets of Ω with pairwise distinct sizes to a list of subsets of Ω as follows. Let A be any such set. There exists $m \in \mathbb{N} \cup \{0\}$ and subsets $A_1, \ldots, A_m$ of Ω, uniquely indexed by increasing size, such that $A = \{A_1, \ldots, A_m\}$. We define $\sigma(A) := [A_1, \ldots, A_m]$, and remark that a permutation of Ω stabilises the set of subsets A

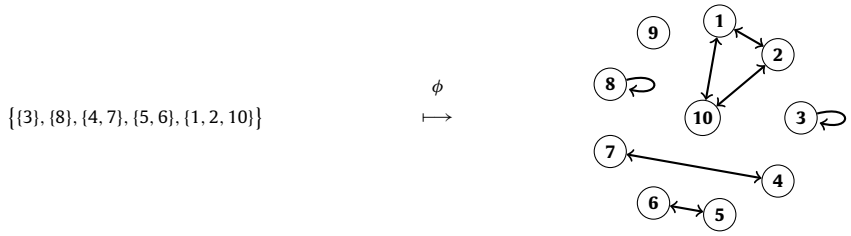

Fig. 2. An example of the function ϕ from Section 8.2.2 for the given set of disjoint subsets of $\Omega := \{1, \ldots, 10\}$. To reduce visual clutter, loops are omitted at vertices of non-singleton subsets.

if and only if it stabilises each of the subsets that it contains. Perfect refiners for the stabilisers and transporters can thus be obtained from σ via Section 8.1.1, Lemma 5.5, and Theorem 5.3.

8.2. Examples of perfect refiners in graph backtracking

For the kinds of objects listed by graph backtracking in Table 2, it is easy to find examples for all $|\Omega| \geq 4$ where the stabiliser in $\mathrm{Sym}(\Omega)$ is not a direct product of symmetric groups, which means that the stabiliser has no perfect refiner in partition backtracking (see Table 1). It remains to show that there are perfect refiners in graph backtracking for the stabilisers and transporter sets of these objects.

8.2.1. Graph and digraph automorphisms and isomorphisms

We recall our standard notation for graphs from Section 2. Let π be the function that maps each graph (Ω, E) to the digraph $(\Omega, \{(\alpha, \beta), (\beta, \alpha) : \{\alpha, \beta\} \in E\})$. This means that each edge $\{\alpha, \beta\}$ is replaced by the arcs (α, β) and (β, α). In addition, we define x to be some fixed label, and we let σ be the function that maps each digraph on Ω to the labelled digraph on Ω formed by assigning the label x to all of its vertices and arcs. Then π and σ are injective $\mathrm{Sym}(\Omega)$-invariant functions, and so Theorem 5.3 and Corollary 5.1 give perfect refiners for all stabilisers and transporter sets in $\mathrm{Sym}(\Omega)$ of graphs and digraphs on Ω in graph backtracking.

8.2.2. Stabilisers and transporters of sets of nonempty pairwise disjoint sets

We define a function ϕ from the set of all subsets of Ω to the set of all digraphs on Ω as follows. Let $A := \{A_1, \ldots, A_k\}$, for some $k \in \mathbb{N} \cup \{0\}$, be a set of subsets of Ω, and define $\phi(A)$ to be the digraph on Ω that consists of a clique (plus loops) on the vertices of each member of A, i.e.

$$\phi(A) := \big(\Omega,\ \{(\alpha, \beta) \in \Omega \times \Omega : \{\alpha, \beta\} \subseteq A_i \text{ for some } i \in \{1, \ldots, k\}\}\big).$$

See Fig. 2 for an example. Then ϕ is $\mathrm{Sym}(\Omega)$-invariant, and so by Theorem 5.3 and Section 8.2.1, we obtain refiners in graph backtracking for the stabilisers and transporter sets of all sets of subsets of Ω. Furthermore, when restricted to sets of nonempty pairwise disjoint subsets of Ω, the function ϕ is injective and $\mathrm{Sym}(\Omega)$-invariant, and so the corresponding refiners are perfect.

8.2.3. Permutation centraliser and conjugacy, and subgroup centraliser

We consider $\mathrm{Sym}(\Omega)$ acting on itself by conjugation. Let ψ be the function from $\mathrm{Sym}(\Omega)$ to the set of all digraphs on Ω where for each $g \in \mathrm{Sym}(\Omega)$, $\psi(g) := \big(\Omega, \{(\alpha, \beta) \in \Omega \times \Omega : \alpha^g = \beta\}\big)$. See Fig. 3, where we explain an example for illustration.

Then ψ is injective and $\mathrm{Sym}(\Omega)$-invariant, giving perfect refiners in graph backtracking for all centralisers in $\mathrm{Sym}(\Omega)$ of elements of $\mathrm{Sym}(\Omega)$, and for the transporter sets for permutation conjugacy (see Theorem 5.3 and Section 8.2.1). By Lemma 5.5, this gives perfect refiners for all stabilisers of lists of permutations in $\mathrm{Sym}(\Omega)$ under conjugation. The centraliser in $\mathrm{Sym}(\Omega)$ of a subgroup of $\mathrm{Sym}(\Omega)$ is the pointwise stabiliser of any of its generating sets, and thus we can obtain a perfect refiner in graph backtracking for the centraliser in $\mathrm{Sym}(\Omega)$ of any subgroup of $\mathrm{Sym}(\Omega)$ given by a generating set.

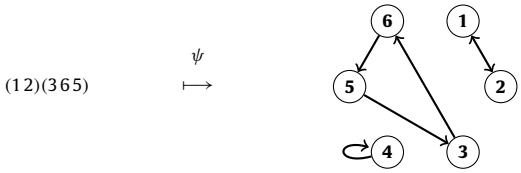

Fig. 3. An example of the function ψ from Section 8.2.3 for the permutation $(1\,2)(3\,6\,5)$, with $\Omega := \{1, \ldots, 6\}$. This digraph gives rise to a perfect refiner for the centraliser of $(1\,2)(3\,6\,5)$ in $\mathrm{Sym}(\Omega)$.

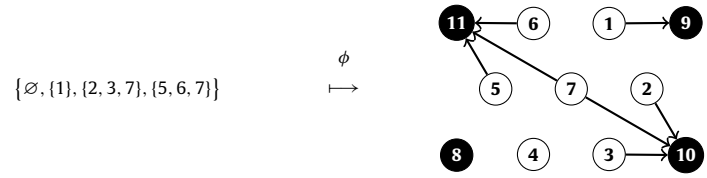

Fig. 4. A representative labelled digraph of the extended graph defined by the function ϕ from Section 8.3.1, for the set of subsets $\{\varnothing, \{1\}, \{2, 3, 7\}, \{5, 6, 7\}\}$ of $\Omega := \{1, \ldots, 7\}$, with $\Lambda := \mathbb{N}$.

8.3. Examples of perfect refiners in extended graph backtracking

By Lemma 7.5, all stabilisers and transporter sets have perfect refiners in extended graph backtracking, but this knowledge is not necessarily immediately useful: the extended graphs underpinning this lemma may require impractically many additional vertices, and anyway, the construction requires knowing the stabiliser or transporter set in advance. In order to avoid too many additional vertices or circular arguments, we wish to construct refiners in extended graph backtracking using only facts about the relevant objects that can be computed cheaply.

In the forthcoming examples, we specify an extended graph by giving one of the labelled digraphs that it contains. Therefore, in each case, we must prove that this is well-defined. The fixed set of vertices Λ is totally ordered, and we always choose vertices from this set in ascending order. Thus it suffices to show, in each case, that our choices lead to labelled digraphs that differ only by a permutation of their vertices in $\Lambda \setminus \Omega$.

8.3.1. Stabilisers and transporters of sets of sets

We define a function ϕ from the set of all sets of subsets of Ω to the set of all extended graphs on Ω. Let $A := \{A_1, \ldots, A_k\}$ be a set of subsets of Ω, where $k \in \mathbb{N} \cup \{0\}$, indexed arbitrarily, and let $V_k := \{\beta_1, \ldots, \beta_k\} \subseteq \Lambda \setminus \Omega$ comprise the k least elements of $\Lambda \setminus \Omega$. We define Γ_A to be the labelled digraph on $\Omega \cup V_k$ with arcs $\bigcup_{i=1}^{k} \{(\alpha, \beta_i) : \alpha \in A_i\}$, where vertices in Ω are labelled *white*, and all other vertices and arcs are labelled *black*. Thus there is a *white* vertex in Γ_A for each member of Ω, and a *black* vertex for each member of A; each vertex in Ω has arcs to the *black* vertices corresponding to the members of A that contain it. We define $\phi(A) := \overline{\Gamma_A}$ to be the extended graph containing Γ_A (Fig. 4).

The only choice involved in constructing a labelled digraph using the method described above is the indexing of the members of A. Therefore the labelled digraphs produced can differ only by a permutation of the names of the vertices outside of Ω, and ϕ is well-defined. It is clear that ϕ is injective and $\mathrm{Sym}(\Omega)$-invariant, and so perfect refiners are given by Theorem 5.3 and Corollary 5.1.

Next, we discuss a situation where there is no perfect refiner in graph backtracking. Let H be the subgroup $\langle (1\,2\,3)(4\,5\,6), (1\,2)(3\,5) \rangle$ of $\mathrm{Sym}(\{1, \ldots, 6\})$. It can be shown that H is a proper subgroup of $\mathrm{Sym}(\{1, \ldots, 6\})$ that acts 2-transitively on Ω, and which is therefore not 2-closed. If we define O to be the orbit of $\{1, 2, 3\}$ under H, then H is exactly the stabiliser of O in $\mathrm{Sym}(\{1, \ldots, 6\})$. Let $n \in \mathbb{N}$ with $n \geq 7$ and let $\Omega := \{1, \ldots, n\}$. It follows that the group $G := H \times \mathrm{Sym}(\{7, \ldots, n\})$ (identifying this

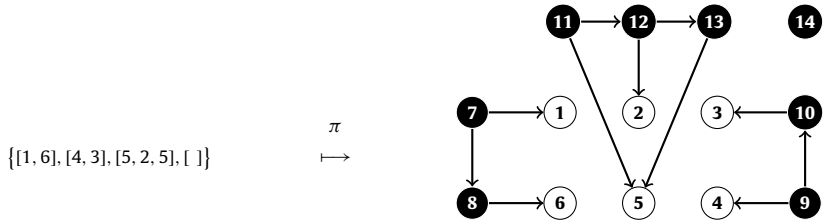

Fig. 5. A representative labelled digraph of the extended graph defined by the function π from Section 8.3.2, for the set of lists $A := \{[1, 6], [4, 3], [5, 2, 5], [\,]\}$ in $\Omega := \{1, \ldots, 6\}$, with $\Lambda := \mathbb{N}$. Its stabiliser in $\mathrm{Sym}(\{1, \ldots, 14)\}$ preserves Ω setwise, and the restriction of this stabiliser to Ω is $\mathrm{Stab}(A)$.

direct product with a subgroup of $\mathrm{Sym}(\Omega)$ as in Table 1) is the stabiliser in $\mathrm{Sym}(\Omega)$ of the set of subsets O, but it is not 2-closed, and therefore it has no perfect refiner in graph backtracking.

8.3.2. Stabilisers and transporters of sets of lists

We define a function π that maps each set of lists in Ω to an extended graph on Ω. Let $A := \{A_1, \ldots, A_k\}$ be a set of nonempty lists in Ω, for some $k \in \mathbb{N} \cup \{0\}$, indexed arbitrarily. Moreover, let $r := \prod_{i=1}^{k} |A_i|$ be the product of the lengths of these lists, let V_r be a set of the least r elements of $\Lambda \setminus \Omega$, and let ρ be an arbitrary bijection from $\{(i, j) : i \in \{1, \ldots, k\} \text{ and } j \in \{1, \ldots, |A_i|\}\}$ to V_r.

We define $\pi(A)$ to be the extended graph that contains the labelled digraph on $\Omega \cup V_r$ with arcs

$$\left\{\left(\rho(i, j),\ A_i[j]\right) : 1 \le i \le k,\ 1 \le j \le |A_i|\right\} \cup \left\{\left(\rho(i, j),\ \rho(i, j+1)\right) : 1 \le i \le k,\ 1 \le j < |A_i|\right\},$$

where the vertices in Ω are labelled *white*, and the vertices in V_r and all arcs are labelled *black*.

Each vertex in V_r corresponds via ρ to a position in a list in A; the arcs between vertices in V_r encode the ordering of each list, and the arcs towards vertices in Ω encode the entry at each position.

For a set of lists in Ω that includes the empty list, [ ], we proceed as above, except that the labelled digraph has an additional isolated vertex (the next least element of $\Lambda \setminus \Omega$) with label *black*.

It is straightforward to see that π is well-defined (once a bijection ρ is fixed), injective, and $\mathrm{Sym}(\Omega)$-invariant, and so with this method, it is possible to use Theorem 5.3 and Corollary 5.1 to produce perfect refiners in extended graph backtracking for the stabilisers and transporter sets of any sets of lists in Ω (Fig. 5).

In combination with Lemma 7.2, the following lemma implies that for all sets Ω with $|\Omega| > 3$, there exist sets of lists in Ω whose stabilisers and transporter sets do not have perfect refiners in graph backtracking. This result also gives a different constructive proof of Lemma 7.5. However, the method described in this section is impractical for constructing a perfect refiner for an arbitrary subgroup $G \le \mathrm{Sym}(\Omega)$, since it would produce a labelled digraph with $(|G| + 1)|\Omega|$ vertices.

Lemma 8.1. *Every subgroup of* $\mathrm{Sym}(\Omega)$ *is the stabiliser in* $\mathrm{Sym}(\Omega)$ *of a set of lists in* Ω.

Proof. Let $G \le \mathrm{Sym}(\Omega)$ and A be an enumeration of Ω. Then G is the stabiliser in $\mathrm{Sym}(\Omega)$ of A^G. □

8.3.3. Stabilisers and transporters of sets of graphs, digraphs, or labelled digraphs

We define a function ψ from the set of all sets of labelled digraphs on Ω to the set of all extended graphs on Ω that is injective and $\mathrm{Sym}(\Omega)$-invariant. Therefore ψ can be combined with the functions of Section 8.2.1, via Theorem 5.3, to give perfect refiners in extended graph backtracking for the stabilisers and transporter sets of sets of graphs or digraphs on Ω that are not necessarily labelled.

First, we fix labels # and ⊛ that are not allowed to be used as labels in any labelled digraph on Ω.

Let $A := \{\Gamma_1, \ldots, \Gamma_k\}$ be a set of labelled digraphs on Ω, for some $k \in \mathbb{N} \cup \{0\}$, indexed arbitrarily.

Roughly speaking, we build a new labelled digraph from a disjoint union of copies of the k members of A, retaining labels, and adding arcs that anchor each copy, and arcs that maintain the

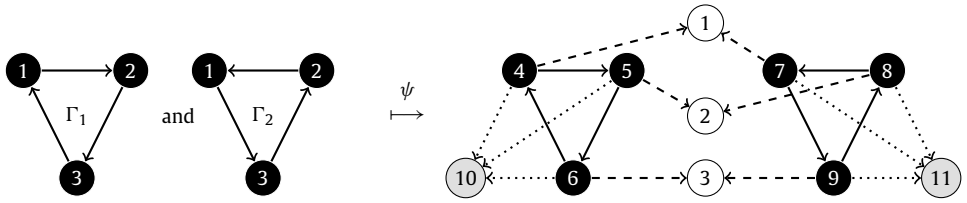

Fig. 6. Labelled digraphs Γ_1 and Γ_2 on $\Omega := \{1, 2, 3\}$ with all labels *black*, and a representative labelled digraph of the extended graph $\psi(\{\Gamma_1, \Gamma_2\})$, with $\Lambda := \mathbb{N}$ and ψ from Section 8.3.3. Vertices and arcs are differently patterned, according to label. The stabiliser in $\mathrm{Sym}(\{1, \ldots, 11\})$ of this labelled digraph preserves Ω setwise, and its restriction to Ω is the setwise stabiliser of $\{\Gamma_1, \Gamma_2\}$ in $\mathrm{Sym}(\Omega)$.

correspondences between Ω and the vertices of the copies. We give an example in Fig. 6 and discuss the precise construction in the following paragraph.

Let $\{V_1, \ldots, V_k\}$ be an arbitrary partition of the least $k|\Omega|$ elements of $\Lambda \setminus \Omega$ into k parts of size $|\Omega|$, and for each $i \in \{1, \ldots, k\}$, let τ_i be an arbitrary bijection from Ω to V_i. In addition, let $X := \{x_1, \ldots, x_k\}$ be the least k elements of $\Lambda \setminus (\Omega \cup V_1 \cup \cdots \cup V_k)$, indexed arbitrarily. Then we define $\psi(A)$ to be the extended graph that contains the labelled digraph on $\Omega \cup V_1 \cup \cdots \cup V_k \cup X$ with arcs

$$\bigcup_{i=1}^{k} (\{(\tau_i(\alpha), \alpha),\ (\tau_i(\alpha), x_i) : \alpha \in \Omega\} \cup \{(\tau_i(\alpha), \tau_i(\beta)) : (\alpha, \beta) \text{ is an arc of } \Gamma_i\}),$$

where vertices in Ω and arcs ending in Ω are labelled #, vertices in X and arcs ending in X are labelled $\circledast$, and for all $\alpha, \beta \in \Omega$ and $i \in \{1, \ldots, k\}$, the vertex $\tau_i(\alpha)$ has the label of α in Γ_i, and the arc $(\tau_i(\alpha), \tau_i(\beta))$ (if present) has the label of (α, β) in Γ_i.

Different choices in the construction lead to labelled digraphs that differ only by a permutation of the vertices in $\Lambda \setminus \Omega$, so ψ is well-defined. The injectivity of ψ is guaranteed by the vertices in X and their arcs; these could be omitted for sets of connected labelled digraphs. It is clear that ψ is $\mathrm{Sym}(\Omega)$-invariant.

Remark 8.2. (a) The proof of Lemma 8.1 can be adapted to show that when $|\Omega| > 3$, there exist stabilisers and transporter sets of sets of labelled digraphs on Ω without perfect refiners in graph backtracking.
(b) The fundamental idea behind this technique can be adapted to build perfect refiners for the stabilisers and transporter sets of extended graphs in extended graph backtracking.

8.3.4. Stabilisers and transporters of sets of labelled digraph stacks

Let ζ be the injective $\mathrm{Sym}(\Omega)$-invariant function from the set of all labelled digraphs stacks on Ω to the set of all labelled digraphs on Ω that is given in Jefferson et al. (2021, Section 3.1). Moreover, let $k \in \mathbb{N}_0$. Then the function that maps any set $\{S_1, \ldots, S_k\}$ of labelled digraph stacks on Ω to the set of labelled digraphs $\{\zeta(S_1), \ldots, \zeta(S_k)\}$ is injective and $\mathrm{Sym}(\Omega)$-invariant. Thus we can use Theorem 5.3 in combination with Section 8.3.3 to construct perfect refiners in extended graph backtracking for the stabilisers and transporter sets of arbitrary sets of labelled digraphs stacks on Ω. This allows perfect refiners in graph backtracking for the stabilisers and transporter sets of one kind of object to be converted into perfect refiners in extended graph backtracking for the stabilisers and transporter sets of sets of those objects.

8.3.5. Normalisers and sets of conjugating elements of subgroups

Any subgroup of a group K acts on the set of subgroups of K by conjugation and the stabiliser of a subgroup G under conjugation by a subgroup H is the *normaliser of G in H*, denoted $N_H(G)$.

Computing normalisers and deciding subgroup conjugacy are typical use cases for backtrack search, but existing techniques seem to find these problems particularly difficult. Describing refiners for normalisers has been and continues to be an important area of research, see for example Theißen (1997), Chang (2021).

One intuitive explanation for the difficulty is that a normaliser may permute structures of a group (such as orbits) that would be fixed in the context of other search problems, which may reduce the potential for pruning. In the earlier parts of Section 8.3 and in Table 2, we have seen constructions of perfect refiners in extended graph backtracking for the stabilisers and transporter sets in $\mathrm{Sym}(\Omega)$ of many kinds of *sets* of objects that do not have perfect refiners in existing backtracking settings.

This suggests that the extended graph backtracking technique may lend itself well to the development of better refiners for normalisers and for sets of conjugating elements of subgroups. The strategy would be to identify structures that are permuted by the normaliser, and to then develop refiners for the stabilisers and transporter sets of sets of such structures. As a first step, we examine orbital graphs.

Proposition 8.3. *Let $G, H \leq \mathrm{Sym}(\Omega)$ and $x \in \mathrm{Sym}(\Omega)$. If $G^x = H$, then x transports the set of orbital graphs of G to the set of orbital graphs of H. In particular, the normaliser of G in $\mathrm{Sym}(\Omega)$ stabilises the set of orbital graphs of G.*

Proof. It is routine to verify that if $G^x = H$, then for all $\alpha, \beta \in \Omega$, x transports the orbital graph of G with base-pair (α, β) to the orbital graph of H with base-pair (α^x, β^x). □

Proposition 8.3 implies that the function μ that maps a subgroup of $\mathrm{Sym}(\Omega)$ to its set of orbital graphs is $\mathrm{Sym}(\Omega)$-invariant. In Section 8.3.3, we described perfect refiners in extended graph backtracking for the stabilisers and transporter sets of arbitrary sets of digraphs, and so by Theorem 5.3, we can use μ to obtain refiners for the normalisers and sets of conjugating elements of subgroups. However, if $|\Omega| \geq 4$, then μ is not injective (consider different 2-transitive subgroups of $\mathrm{Sym}(\Omega)$), so these refiners are not necessarily perfect. Nevertheless, μ gives many instances of perfect refiners. For example, it is clear that the set of 2-closed subgroups of $\mathrm{Sym}(\Omega)$ is closed under conjugation by $\mathrm{Sym}(\Omega)$, and that the restriction of μ to this set is injective. Therefore those refiners are perfect.

Furthermore, by Proposition 8.3, the refiner described above for the normaliser of a subgroup $G \leq \mathrm{Sym}(\Omega)$ is perfect if and only if $N_{\mathrm{Sym}(\Omega)}(G)$ is equal to the stabiliser in $\mathrm{Sym}(\Omega)$ of the set of orbital graphs of G. By Lemma 8.4, these are the subgroups whose normaliser in $\mathrm{Sym}(\Omega)$ coincides with the normaliser of its 2-closure. Example 8.5 shows that this includes more than just the 2-closed subgroups of $\mathrm{Sym}(\Omega)$.

Lemma 8.4. *Let $G \leq \mathrm{Sym}(\Omega)$. The stabiliser in $\mathrm{Sym}(\Omega)$ of the set of orbital graphs of G is the normaliser in $\mathrm{Sym}(\Omega)$ of the 2-closure of G.*

Proof. Let K denote the 2-closure of G. The orbital graphs of G and K coincide by definition, and so $N_{\mathrm{Sym}(\Omega)}(G)$ is contained in the stabiliser by Proposition 8.3. Conversely, let $x \in \mathrm{Sym}(\Omega)$ stabilise the set of orbital graphs of G (and K) and let $g \in K$. For all orbital graphs Γ of K, $\Gamma^{x^{-1}}$ is an orbital graph of K by assumption, and $\Gamma^{x^{-1}g} = \Gamma^{x^{-1}}$ since $g \in K$. Hence $\Gamma^{x^{-1}gx} = \Gamma$, and $x^{-1}gx \in K$. □

Example 8.5. Let $\Omega := \{1, \ldots, 6\}$ and let $G := \langle (1\,2\,3)(4\,5\,6), (1\,4)(2\,5) \rangle \leq \mathrm{Sym}(\Omega)$. The orbital graphs of G are depicted in Fig. 7. The normaliser of G in $\mathrm{Sym}(\Omega)$ is the stabiliser in $\mathrm{Sym}(\Omega)$ of the set of orbital graphs of G, even though G is not 2-closed. We note that $G < N_{\mathrm{Sym}(\Omega)}(G) < \mathrm{Sym}(\Omega)$.

9. Closing remarks

We have introduced the concept of perfect refiners, which gives a way of comparing the available pruning power in the various backtracking frameworks and which, within a framework, gives a way of comparing refiners for a given set. This is naturally complemented by the introduction of extended graph backtracking. We have also discussed the existence of perfect refiners in the different frameworks, and given concrete examples of perfect refiners that are implemented in Vole.

This work suggests several obvious questions and directions for further investigation.

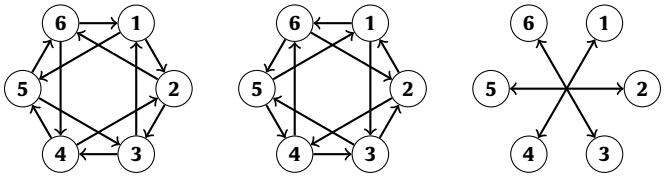

Fig. 7. The orbital graphs of the subgroup G of $\mathrm{Sym}(\{1, \ldots, 6\})$ from Example 8.5.

For any given search problem, which backtracking framework is the most appropriate for solving it, and how should this decision be made? Having decided upon a framework, which refiners should be used? In which other ways can we compare and understand refiners?

One obvious line of inquiry is the development of further methods for comparing different refiners for the same set, and of measuring the 'quality' of a given refiner. The notion of perfectness is binary, and it fails to capture any nuance in the way that a refiner may fail to be perfect.

It would also be useful to better understand the performance implications of using a given refiner in a backtrack search algorithm. Roughly speaking, a perfect refiner for a given set is best possible in terms of search size. On the other hand, like any refiner, a perfect refiner may be impractically expensive to compute with, and this might partially or fully override the search-size advantage. For refiners in extended graph backtracking, we could begin to understand the interplay between these effects by distinguishing refiners according to how many vertices and arcs their extended graphs require.

We have seen in Proposition 7.4 and Lemma 7.5 that every group and coset has a perfect refiner in extended graph backtracking, but that the digraphs in Bouwer (1969), Kearnes (2015) that underpin these results may be impractical for computation. Extended graph backtracking would therefore benefit from a better understanding of the theoretical and practical possibilities for representing groups by digraphs of the kind in Proposition 7.4, because then we could work on finding perfect refiners, or decide that that is infeasible.

As mentioned in Section 8.3.5, extended graph backtracking appears to be well suited to the development of better refiners for use in normaliser and subgroup conjugacy computations. This is already an active area of research, motivated partly by the fact that good algorithms for normaliser search exist (e.g. in MAGMA, see Bosma et al., 1997). Our work in this direction will be continued in the future.

Finally, we note that there is ongoing work to apply the ideas of the graph backtracking and extended graph backtracking frameworks to canonisation algorithms in finite symmetric groups.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Bosma, Wieb, Cannon, John, Playoust, Catherine, 1997. The Magma algebra system. I. The user language. J. Symb. Comput. 24 (3–4), 235–265. Computational algebra and number theory, London, 1993.

Bouwer, I.Z., 1969. Section graphs for finite permutation groups. J. Comb. Theory 6, 378–386.

Chang, Mun See, 2021. Computing normalisers of highly intransitive groups. PhD thesis. University of St Andrews.

Chang, Mun See, Jefferson, Christopher, Wilson, Wilf A., 2021. Vole – GAP Package, Version 0.5.2.

Dixon, John D., Mortimer, Brian, 1996. Permutation Groups. Graduate Texts in Mathematics, vol. 163. Springer-Verlag, New York.

Jefferson, Christopher, Pfeiffer, Markus, Waldecker, Rebecca, 2019a. New refiners for permutation group search. J. Symb. Comput. 92, 70–92.

Jefferson, Christopher, Pfeiffer, Markus, Waldecker, Rebecca, Wilson, Wilf A., 2019. Permutation group algorithms based on directed graphs (extended version). Preprint. arXiv:1911.04783.

Jefferson, Christopher, Pfeiffer, Markus, Wilson, Wilf A., Waldecker, Rebecca, 2021. Permutation group algorithms based on directed graphs. J. Algebra 585, 723–758.

Kearnes, Keith, 2015. Can every permutation group be realized as the automorphism group of a graph (acting on a subset of the vertices)? MathOverflow. https://mathoverflow.net/q/215472 (version: 2015-08-23).

Leon, Jeffrey S., 1991. Permutation group algorithms based on partitions. I. Theory and algorithms. J. Symb. Comput. 12 (4–5), 533–583. Computational group theory, part 2.

Leon, Jeffrey S., 1997. Partitions, refinements, and permutation group computation. In: Groups and Computation, II. New Brunswick, NJ, 1995. In: DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 28. Amer. Math. Soc., Providence, RI, pp. 123–158.

Sims, Charles C., 1971. Computation with permutation groups. In: Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation. SYMSAC'71. ACM, New York, NY, USA, pp. 23–28.

Theißen, Heiko, 1997. Eine Methode zur Normalisatorberechnung in Permutationsgruppen mit Anwendungen in der Konstruktion primitiver Gruppen. Verlag der Augustinus-Buchh.