Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Exploiting Symmetries by Planning for a Descriptive Quotient

Mohammad Abdulaziz[†] and **Michael Norrish**[†] and **Charles Gretton**[†‡]
[†]Canberra Research Lab., NICTA[*], [†]Australian National University, [‡]Griffith University

Abstract

We eliminate symmetry from a problem before searching for a plan. The planning problem with symmetries is decomposed into a set of isomorphic subproblems. One plan is computed for a small planning problem posed by a *descriptive quotient*, a description of any such subproblem. A concrete plan is synthesized by concatenating instantiations of that one plan for each subproblem. Our approach is sound.

1 Introduction

The planning and model checking communities have for some time sought methods to exploit symmetries that occur in transition systems. The quintessential planning scenario which exhibits symmetries is GRIPPER. This comprises a robot whose left and right grippers can be used interchangeably in the task of moving a set of N indistinguishable packages from a source location to a goal location. Intuitively the left and right grippers are symmetric because if we changed their names, by interchanging the terms left and right in the problem description, we are left with an identical problem. Packages are also interchangeable and symmetric.

One method to exploit symmetry is to perform checking of properties of interest—*e.g.*, goal reachability—in a *quotient* system, which corresponds to a (sometimes exponentially) smaller bisimulation of the system at hand. Wahl and Donaldson (2010) review the related model checking literature. Such methods were recently adapted for planning and studied by Pochter *et al.* (2011), Domshlak *et al.* (2012; 2013) and Shleyfman *et al.* (2015). Related work about state-based planning in equivalence classes of symmetric states includes [Guere and Alami, 2001; Fox and Long, 1999; 2002].

Planning in a quotient system, a state s is represented by a *canonical* element from its *orbit*, the set of states which are symmetric to s. Giving integer labels to packages in GRIPPER, when the search encounters a state where the robot is holding 1 package using 1 gripper, this is represented using the canonical state where, for example, the left gripper is holding package with identity "1". Orbit search explores the quotient system by simulating actions from known canonical states, and then computing the canonical representations of resultant states. That canonicalisation step requires the solution to the *constructive orbit problem* [Clarke *et al.*, 1998] which is *NP-hard* [Eugene, 1993]. A key weakness, is that for each state encountered by the search an intractable canonicalisation operation is performed. This is mitigated in practice by using approximate canonicalisation. By forgoing exact canonicalisation, one encounters a much larger search problem than necessary. For a GRIPPER instance with 42 packages, the breadth-first orbit search with approximate canonicalisation by Pochter *et al.* reportedly performs $60.5K$ state expansion operations, far more than necessary.

Following the seminal work by Crawford *et al.* (1996) and Brown *et al.* (1996), when planning *via* constraint satisfaction, known symmetries are exploited by: (i) including symmetry breaking constraints, either directly as part of the problem expression [Joslin and Roy, 1997; Rintanen, 2003], or (ii) otherwise dynamically as suggested by [Miguel, 2001] as part of a *nogood* mechanism. In GRIPPER, we can statically require that if no package is yet retrieved, then any retrieval targets package 1 with the left gripper. Dynamically, having proved that no 3-step plan exists retrieving package 1 with the left gripper, then no plan of that length exists retrieving package $i \neq 1$ using either gripper. Searching using the proposed dynamic approach is quite weak, as symmetries are only broken as nogood information becomes available. A weakness of both approaches is that problems expressed as CSPs include variables describing the state of the transition system at different plan steps. Existing approaches do not break symmetries across steps, and can therefore waste effort exploring partial assignments that express executions which visit symmetric states at different steps.

Our contribution is a novel procedure for domain-independent planning with symmetries. Following, *e.g.*, Pochter *et al.*, in a first step we infer knowledge about problem symmetries from the description of the problem. Then departing from existing approaches, our second step uses that knowledge to obtain a quotient of the concrete problem description. Called the *descriptive quotient*, this describes any element in the set of isomorphic subproblems which abstractly model the concrete problem. Third, we invoke a planner once to solve the small problem posed by that descrip-

[*]NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

tive quotient. In the fourth and final step, a concrete plan is synthesized by concatenating instantiations of that plan for each isomorphic subproblem. The descriptive quotient of the aforementioned 42-package GRIPPER instance is solved by breadth-first search expanding 6 states. Using our approach, a concrete plan is obtained in under a second. For comparison, LAMA [Richter and Westphal, 2010] takes about 28 seconds.

The non-existence of a plan for the descriptive quotient does not exclude the possibility of a plan for the concrete problem. Although sound, in that respect our approach is incomplete. Having an optimal plan for the quotient does not guarantee optimality in the concatenated plan. Aside from computing a plan for the descriptive quotient, the computationally challenging steps occur in preprocessing:

(i) Identification of problem symmetries from the original description, a problem as hard as graph isomorphism, which is not known to be tractable, and

(ii) Computing an appropriate set of sub-problems isomorphic to the quotient.

2 Definitions and Notations

We formally define *planning problems* and related concepts. Where we do not refer to a specific function, we shall use the symbol f. We formalise functions as sets of key-value pairs $(k \mapsto v)$. We use the term *domain* mathematically. We write $\mathcal{D}(f)$ for the domain of f, *i.e.*, $\{k \mid (k \mapsto v) \in f\}$. We write $\mathcal{R}(f)$ for the range of f, *i.e.*, $\{v \mid (k \mapsto v) \in f\}$.

Definition 1 (States and Actions). *A planning problem is defined in terms of* states *and* actions*:*

(i) States are finite maps from variables—i.e., state-characterizing propositions—to Booleans.

(ii) An action a is a pair of finite maps over subsets of those variables. The first component of the pair, written $\mathsf{pre}(a)$*, is the* precondition *and the second component of the pair (*$\mathsf{eff}(a)$*) is the* effect*. The* domain *of an action is the union of the domains of the two finite maps. For a set of actions A, we define the set of preconditions* $\mathsf{pre}(A)$ *as* $\bigcup_{a \in A} \mathsf{pre}(a)$.

We give examples of states and actions using sets of literals. For example, $\{x, \overline{y}, z\}$ is the state where state variables x and z are (map to) true, and y is false.

Definition 2 (Planning Problem). *A planning problem Π is a 3-tuple $\langle I, A, G \rangle$, with I the initial state of the problem, G a description of goal states (another finite map from variables to Booleans), and A a set of actions. We define the domain of the problem ($\mathcal{D}(\Pi)$) to be domain of the initial state ($\mathcal{D}(I)$). Problem Π is* valid *if $\mathcal{D}(G) \subseteq \mathcal{D}(I)$, and all actions refer exclusively to variables that occur in $\mathcal{D}(I)$. We only consider valid problems. Hereafter we refer to the initial state, actions or goal of problem Π as $\Pi.I$, $\Pi.A$ or $\Pi.G$ respectively. We may also omit the Π if it is clear from the context, e.g. I for $\Pi.I$ and A_i for $\Pi_i.A$. A state s is valid with respect to a planning problem Π if $\mathcal{D}(s) = \mathcal{D}(I)$.*

Definition 3 (Action Execution and Plan). *When an action a is executed at state s, written $\mathsf{e}(s, a)$, it produces a successor state s'. If $\mathsf{pre}(a) \not\subseteq s$, then $s' = s$. Otherwise s' is a valid state where $\mathsf{eff}(a) \subseteq s'$ and $s(x) = s'(x)$ $\forall x \in \mathcal{D}(I) \setminus \mathcal{D}(\mathsf{eff}(a))$. We lift e to sequences of executions, taking an action sequence $\dot{\pi}$ as the second argument. So $\mathsf{e}(s, \dot{\pi})$ denotes the state resulting from successively applying each action from $\dot{\pi}$ in turn, starting from s. An action sequence is a plan/solution if its execution from I yields a goal state.*

Definition 4 (Subproblem). *Problem Π_1 is a subproblem of Π_2, written $\Pi_1 \subseteq \Pi_2$, if $\mathcal{D}(I_1) \subseteq \mathcal{D}(I_2)$, and if $A_1 \subseteq A_2$.*

Definition 4 purposefully does not consider problem goals. We later consider subproblems with a variety of goals which do not all correspond to concrete problem goals. We form a concrete plan by concatenating plans for subproblems with such varieties of goals. For example, we will consider subproblems from GRIPPER with only one package and one gripper, where the goal is to: (i) relocate that package according to the concrete goal, (ii) additionally free the gripper, and (iii) additionally have the robot relocate to its starting position. In this case a concrete plan is formed by concatenating plans for subproblems given for each distinct package.

We require a few additional notations. Let m be a finite map, *e.g.*, a state s, the assignments $\mathsf{pre}(a)$, *etc.* Then, let $f(\!|m|\!)$ be the image of m under function f: the map $\{f(k) \mapsto v \mid (k \mapsto v) \in m\}$. This is well-defined if $f(k_1) = f(k_2)$ implies that $m(k_1) = m(k_2)$. We lift this notion of image to other composite types. For example, we write $f(\!|\Pi|\!)$ for the image of Π under f, where all finite maps in Π are transformed by f.

3 Computing Problem Symmetries

To exploit problem symmetries we must first discover them. We follow the discovery approach from [Pochter *et al.*, 2011], restricting ourselves to Boolean-valued variables in $\mathcal{D}(\Pi)$. We assume familiarity with *groups, subgroups, group actions*, and *graph automorphism groups*. Symmetries in a problem description are defined by an *automorphism group*.

Definition 5 (Problem Automorphism Group). *The automorphism group of Π is: $Aut(\Pi) = \{\sigma \mid \sigma(\!|\Pi|\!) = \Pi\}$. Members of $Aut(\Pi)$ are permutations on $\mathcal{D}(\Pi)$.*

A graphical representation of Π is constructed so vertex symmetries in it correspond to variable symmetries in Π. We follow the graphical representation introduced in [Pochter *et al.*, 2011].

Definition 6 (Problem Description Graph (PDG)). *The (undirected) graph $\Gamma(\Pi)$ for a planning problem Π, is defined as follows:*

(i) $\Gamma(\Pi)$ has two vertices, $v^\top$ and $v^\bot$, for every variable $v \in \mathcal{D}(\Pi)$; two vertices, a_p and a_e, for every action $a \in A$; and vertex v_I for I and v_G for G.

(ii) $\Gamma(\Pi)$ has an edge between $v^\top$ and $v^\bot$ for all $v \in \mathcal{D}(\Pi)$; between a_p and a_e for all $a \in A$; between a_p and v^* if $(v \mapsto *) \in \mathsf{pre}(a)$, and between a_e and v^* if $(v \mapsto *) \in \mathsf{eff}(a)$; between v^* and v_I if $(v \mapsto *)$ occurs in I; and between v^* and v_G if $(v \mapsto *)$ occurs in G.

We write $V(\Gamma)$ for the set of vertices, and $E(\Gamma)$ for edges of a graph Γ.

The automorphism group of the PDG, $Aut(\Gamma(\Pi))$, is identified by solving an undirected graph isomorphism problem. The action of a subgroup of $Aut(\Gamma(\Pi))$ on $V(\Gamma(\Pi))$ induces a *partition*, called the *orbits*, of $V(\Gamma(\Pi))$. We can now define our *quotient* structures based on partitions $\mathcal{P}$ of $\mathcal{D}(\Pi)$.

Definition 7 (Quotient Problems and Graphs). *Given partition $\mathcal{P}$ of $\mathcal{D}(\Pi)$, let $\mathcal{Q}$ map members of $\mathcal{D}(\Pi)$ to their equivalence class in $\mathcal{P}$. The descriptive quotient is $\Pi/\mathcal{P} = \mathcal{Q}(\!|\Pi|\!)$, the image of Π under $\mathcal{Q}$. This is well-defined if $\mathcal{P}$ is a set of orbits. We assume that quotient problems are well-defined.*

For graph Γ and a partition $\mathcal{P}$ of its vertices, the quotient $\Gamma/\mathcal{P}$ is the graph with a vertex for each $p \in \mathcal{P}$. $\Gamma/\mathcal{P}$ has an edge between any $p_1, p_2 \in \mathcal{P}$ iff Γ has an edge between any $v \in p_1$ and $u \in p_2$.

To ensure correspondence between PDG symmetries and problem symmetries, we must ensure incompatible descriptive elements do not coalesce in the same orbit. For example, we cannot have action precondition symbols and state-variables in the same orbit.

Definition 8 (Well-formed Partitions). *A partition of $V(\Gamma(\Pi))$ is* well-formed *iff:*

(i) Positive ($v^\top$) and negative ($v^\bot$) variable assignment vertices only coalesce with ones of the same parity;

(ii) Precondition (a_p) and effect (a_e) vertices only coalesce with preconditions and effects respectively, and

(iii) Both v_I and v_G are always in a singleton.

A well-formed partition $\hat{\mathcal{P}}$ defines a corresponding partition $\mathcal{P}$ of $\mathcal{D}(\Pi)$, so that $\Gamma(\Pi)/\hat{\mathcal{P}}$ is isomorphic to $\Gamma(\Pi/\mathcal{P})$.

To ensure well-formedness, vertex symmetry is calculated using the *coloured* graph-isomorphism procedure (CGIP). Vertices of distinct colour cannot coalesce in the same orbit. Vertices of $\Gamma(\Pi)$ are coloured to ensure the orbits correspond to a well-formed partition.

4 Computing the Set of Instantiations

Recapping, symmetries in Π are the basis of a partition $\mathcal{P}$ of its domain $\mathcal{D}(\Pi)$ into orbits. That exposes the *descriptive quotient*, $\Pi/\mathcal{P}$, an abstract problem whose variables correspond to orbits of concrete symbols. Our task now is to compute a set of *instantiations* of the quotient which *cover* all the goal variables $\mathcal{D}(G)$. Called a covering set of isomorphic subproblems, by instantiating a quotient plan for each subproblem and concatenating the results we intend to arrive at a concrete plan. We describe a pragmatic approach to obtaining that covering set. We establish that a covering set is not guaranteed to exists, and give an approach to *refining* partitions to mitigate that fact. We derive a theoretical bound on the necessary size of a covering set, and prove that a general graph formulation of the problem of computing a covering instantiation is NP-complete.

Definition 9 (Transversals, Instantiations). *A transversal $\pitchfork$ is an injective choice function over $\mathcal{P}$ such that $\pitchfork(c) \in c$ for every equivalence class $c \in \mathcal{P}$. A transversal* covers *v if $v \in \mathcal{R}(\pitchfork)$. If $\mathcal{P}$ is a partition of $\mathcal{D}(\Pi)$, we refer to a transversal $\pitchfork$ of $\mathcal{P}$ as an* instantiation *of $\Pi/\mathcal{P}$. An instantiation $\pitchfork$ is* consistent *with a concrete problem Π if $\pitchfork(\!|\Pi/\mathcal{P}|\!) \subseteq \Pi$. When we use the term "problem" discussing an instantiation $\pitchfork$, we intend $\pitchfork(\!|\Pi/\mathcal{P}|\!)$. Note that $\mathcal{D}(\pitchfork(\!|\Pi/\mathcal{P}|\!)) = \mathcal{R}(\pitchfork)$.*

Example 1. *Consider the set $\{a, b, c, d, e, f\}$, and the equivalence classes $c_1 = \{a, b\}$, $c_2 = \{c, d\}$ and $c_3 = \{e, f\}$ of its members. For the partition $\mathcal{P} = \{c_1, c_2, c_3\}$, $\pitchfork_1 = \{c_1 \mapsto a, c_2 \mapsto c, c_3 \mapsto e\}$ and $\pitchfork_2 = \{c_1 \mapsto b, c_2 \mapsto c, c_3 \mapsto f\}$ are two transversals.*

For each goal variable in the problem, our approach shall need to find a consistent instantiation of the quotient which covers that. We thus face the following problem.

Problem 1. *Given Π and a partition $\mathcal{P}$ of $\mathcal{D}(\Pi)$, is there an instantiation of $\Pi/\mathcal{P}$ consistent with Π that covers a variable $v \in \mathcal{D}(\Pi)$?*

Example 2. *Consider the planning problem Π_2 with*

$$\begin{aligned}\Pi_2.I &= \{x, a, b, m, n\}\\ \Pi_2.A &= \{(\{x, a\}, \{\overline{m}, \overline{x}\}), (\{x, b\}, \{\overline{n}, \overline{x}\}), (\{\}, \{x\})\}\\ \Pi_2.G &= \{\overline{m}, \overline{n}\}\end{aligned}$$

Let $\mathcal{P}$ be $\{p_1 = \{x\}, p_2 = \{a, b\}, p_3 = \{m, n\}\}$. Therefore

$$\begin{aligned}(\Pi_2/\mathcal{P}).I &= \{p_1, p_2, p_3\}\\ (\Pi_2/\mathcal{P}).A &= \{(\{p_1, p_2\}, \{\overline{p_3}, \overline{p_1}\}), (\{\}, \{p_1\})\}\\ (\Pi_2/\mathcal{P}).G &= \{\overline{p_3}\}\end{aligned}$$

Let instantiation $\pitchfork$ be $\{p_1 \mapsto x, p_2 \mapsto a, p_3 \mapsto m\}$. Then, $\pitchfork$ covers m, and $\pitchfork(\!|\Pi_2/\mathcal{P}|\!)$ has $I = \{x, a, m\}$, $A = \{(\{x, a\}, \{\overline{m}, \overline{x}\}), (\{\}, \{x\})\}$ and $G = \{\overline{m}\}$. Thus, $\pitchfork(\!|\Pi_2/\mathcal{P}|\!) \subseteq \Pi_2$, making $\pitchfork$ a consistent instantiation and so a solution to Problem 1 with inputs Π_2, $\mathcal{P}$ and m.

4.1 Finding Instantiations: Practice

We now describe how we obtain a set of isomorphic subproblems of Π that covers the goal variables. We compute a set of instantiations, Δ, of the quotient $\Pi/\mathcal{P}$. Our algorithm first initialises that set of instantiations, $\Delta := \emptyset$. Every iteration of the *main loop* computes a consistent instantiation that covers at least one variable $v \in \mathcal{D}(\Pi.G)$. This is done as follows: we create a new (partial) instantiation $\pitchfork = \{p_v \mapsto v\}$, where p_v is the set in $\mathcal{P}$ containing v. Then we determine whether $\pitchfork$ can be completed, to instantiate every set in $\mathcal{P}$, while being consistent at the same time. This determination is achieved by posing the problem in the theory of uninterpreted functions, and using a satisfiability modulo theory (SMT) solver. Our encoding in SMT is constructive, and if a completion of $\pitchfork$ is possible the solver provides it. If successful, we set $\Delta := \Delta \cup \{\pitchfork\}$ and $G := G \setminus \mathcal{R}(\pitchfork)$, and loop. In the case the SMT solver reports failure, the concrete problem cannot be covered by instantiations of the descriptive quotient, and we report failure. In the worst case of $\mathcal{D}(\Pi.G) = \mathcal{D}(\Pi)$, our *main loop* executes $\Sigma_{p \in \mathcal{P}}|p| - |\mathcal{P}| + 1$ times, and hence we have that many instantiations.

Scenarios need not admit a consistent instantiation.

Example 3. *Take* Π_3 *with*

$$\begin{aligned}\Pi_3.I &= \{\overline{m}, \overline{n}, l\}\\ \Pi_3.A &= \{(\{\overline{m}\}, \{n, \overline{l}\}), (\{\overline{n}\}, \{m, \overline{l}\})\}\\ \Pi_3.G &= \{\overline{l}, \overline{m}, \overline{n}\}\end{aligned}$$

Let $\mathcal{P} = \{p_1 = \{m, n\}, p_2 = \{l\}\}$. *The quotient* $\Pi_3/\mathcal{P}$ *has*

$$\begin{aligned}(\Pi_3/\mathcal{P}).I &= \{\overline{p_1}, p_2\}\\ (\Pi_3/\mathcal{P}).A &= \{(\{\overline{p_1}\}, \{p_1, \overline{p_2}\})\}\\ (\Pi_3/\mathcal{P}).G &= \{\overline{p_1}, \overline{p_2}\}\end{aligned}$$

There are no consistent instantiations of $\Pi_3/\mathcal{P}$ *because* m *and* n *are in the same equivalence class and also occur together in the same action.*

Our example demonstrates a common scenario in the IPC benchmarks, where a partition $\mathcal{P}$ of $\mathcal{D}(\Pi)$ does not admit a consistent instantiation because variables that occur in the same action coalesce in the same member of $\mathcal{P}$. We resolve this situation by refining the partition produced using CGIP to avoid having such variables coalesce in the same orbit. For a $p \in \mathcal{P}$, consider the graph $\Gamma(p)$ with vertices p and edges $\{\{x, y\} \mid \exists a \in \Pi.A \wedge \{x, y\} \subseteq \mathcal{D}(a) \wedge x \neq y\}$. The *chromatic number* N of $\Gamma(p)$ gives us the number of colours needed to colour the corresponding vertices $\hat{p} \in \hat{\mathcal{P}}$ in the PDG. Where two variables occur in the same action, their vertices in $\hat{p}$ are coloured differently. For every $p \in \mathcal{P}$, we use an SMT solver to calculate the required chromatic numbers N and graph colourings for $\Gamma(p)$, then we colour the corresponding vertices in the PDG according to the computed N-colouring of $\Gamma(p)$. Lastly, we again pass the PDG to a CGIP after it is recoloured. In 4 benchmark sets the thus-revised partition admits a consistent instantiation where the initial partition does not. Although the *chromatic number* problem is NP-complete, this step is not a bottleneck in practice because the size of the instances that need to be solved is bounded above by the size of the largest variables orbit.

4.2 Finding Instantiations: Theory

We first develop a theoretical bound on the number of required instantiations—by treating abstract covering transversals—that is tight compared to our pragmatic solution above. To characterise the complexity of our instantiation problem, we then study the general problem of computing covering transversals.

Theorem 1 (B. McKay, 2014). *Let* $\mathcal{G}$ *be a group acting on a set* V *(e.g.,* $\mathcal{D}(\Pi)$*). Suppose* $\pitchfork$ *is a transversal of* O*, a set of orbits induced by the action of* $\mathcal{G}$ *on* V*. Take* $S = \Sigma_{o \in O}|o|$ *and* $M = \max_{o \in O}|o|$*. Where* $\bullet$ *is composition, there will always be a set of transversals* $\mathcal{T}$ *with size* $\leq M \ln(S)$ *such that*

(i) Each element $\pitchfork' \in \mathcal{T}$ *satisfies* $\pitchfork' = \sigma \bullet \pitchfork$ *for some* $\sigma \in \mathcal{G}$*, and*

(ii) For every $v \in V$*, it has an element* $\pitchfork'$ *that covers* v*.*

Proof. Take $N = M \ln(S)$ and let H be a subset of $\mathcal{G}$ obtained by drawing N permutations of V independently at random with replacement. For any orbit $o \in O$, the probability for a $v \in o$ is not in $\mathcal{R}(\sigma \bullet \pitchfork)$ for a randomly drawn $\sigma \in H$ is $1 - 1/|o|$. Let $\mathcal{T} = \{\sigma \bullet \pitchfork \mid \sigma \in H\}$ and $\hat{R} = \bigcup(\mathcal{R}(\!|\mathcal{T}|\!))$. Drawing N times from $\mathcal{G}$ to construct H, the probability that $v \notin \hat{R}$ is $(1 - 1/|o|)^N$. Consider the random quantity $Z = |V \setminus \hat{R}|$ with expected value $\mathbb{E}(Z) = \Sigma_{o \in O}|o|(1-1/|o|)^N$. Since $1-x < e^{-x}$ for $x > 0$, we obtain $\mathbb{E}(Z) < \Sigma_{o \in O}|o|e^{(-N/|o|)} \leq Se^{-N/M} = Se^{-\ln(S)}$. From $xe^{-\ln(x)} = 1$, it follows that $\mathbb{E}(Z) < 1$. Since $Z \in \mathbb{N}$, then probability $Z = 0$ is more than 0, and thus N transversals of O suffice to cover V. □

We conjecture that a much smaller number of transversals is actually required, and in all our experimentation have found that the following conjecture is not violated:

Conjecture 1. *Let* $\mathcal{G}$ *be a group acting on a set* V *(e.g.,* $\mathcal{D}(\Pi)$*). Suppose* $\pitchfork$ *is a transversal of* O*, a set of orbits induced by the action of* $\mathcal{G}$ *on* V*. Take* $M = \max_{o \in O}|o|$*. Where* $\bullet$ *is composition, there will always be a set of transversals* $\mathcal{T}$ *with size* $\leq M$ *such that*

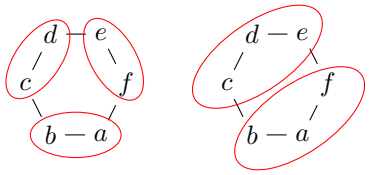
(i) Each element $\pitchfork' \in \mathcal{T}$ *satisfies* $\pitchfork' = \sigma \bullet \pitchfork$ *for some* $\sigma \in \mathcal{G}$*, and*

(ii) For every $v \in V$*, it has an element* $\pitchfork'$ *that covers* v*.*

We are left to formulate and study a general problem of instantiation, treating it as one of finding graph transversals.

Definition 10 (Consistent Graph Transversals). *For a graph* Γ *and a partition* $\mathcal{P}$ *of* $V(\Gamma)$*, a transversal* $\pitchfork$ *of* $\mathcal{P}$ *is consistent with* Γ *when an edge between* p_1 *and* p_2 *in* $E(\Gamma/\mathcal{P})$ *exists iff there is an edge between* $\pitchfork(p_1)$ *and* $\pitchfork(p_2)$ *in* $E(\Gamma)$*.*

Example 4. *Take* Γ *to be the hexagon that we have illustrated twice below in order to depict partitions* $\mathcal{P}_1 = \{\{a, b\}, \{c, d\}, \{e, f\}\}$ *and* $\mathcal{P}_2 = \{p' = \{a, b, f\}, p'' = \{c, d, e\}\}$ *on the left and right, respectively.*



The vertices of $\Gamma/\mathcal{P}_1$ *(a 3-clique) and* $\Gamma/\mathcal{P}_2$ *(a 2-clique) are indicated above by red outlines. There is no consistent transversal of* $\mathcal{P}_1$ *(LHS) because there is no 3-clique subgraph of* Γ *with one vertex from each set in* $\mathcal{P}_1$*. For* $\Gamma/\mathcal{P}_2$ *(RHS),* $\pitchfork_1 = \{p' \mapsto f, p'' \mapsto e\}$ *is a transversal of* $\mathcal{P}_2$ *consistent with* Γ*, because the subgraph of* Γ *induced by* $\{e, f\}$ *is a 2-clique with one vertex from each of* p' *and* p''*.*

A transversal of a well-formed $\hat{\mathcal{P}}$ consistent with $\Gamma(\Pi)$ is isomorphic to an instantiation of $\Pi/\mathcal{P}$ consistent with Π. Accordingly, Problem 1 is an instance of the following.

Problem 2. *Given* Γ*, a partition* $\mathcal{P}$ *of* $V(\Gamma)$ *and* $v \in V(\Gamma)$*, is there a consistent transversal of* $\mathcal{P}$ *which covers* v*?*

We now derive the complexity of Problem 2. We first show that the following problem is NP-complete, and use that result to show that Problem 2 is also NP-complete.

Problem 3. *Given graph* Γ *and a partition* $\mathcal{P}$ *of* $V(\Gamma)$*, is there a transversal of* $\mathcal{P}$ *consistent with* Γ*?*

Lemma 1. *Problem 3 is NP-complete.*

Proof. Membership in NP is given because a transversal's consistency can clearly be tested in polynomial-time. We then show the problem is NP-hard by demonstrating a polynomial-time reduction from SAT. Consider SAT problems given by formulae φ in conjunctive normal form. Assume every pair of clauses is mutually satisfiable—*i.e.*, for clauses $c_1, c_2 \in \varphi$, for two literals $\ell_1 \in c_1$ and $\ell_2 \in c_2$ we have $\ell_1 \neq \neg\ell_2$ (when this assumption is violated, unsatisfiability can be decided in polynomial-time). Consider the graph $\Gamma(\varphi)$, where $V(\Gamma(\varphi)) = \{v_{\ell c} \mid c \in \varphi,\ \ell \in c\}$, and $E(\Gamma(\varphi)) = \{\{v_{\ell_1 c_1}, v_{\ell_2 c_2}\} \mid c_1, c_2 \in \varphi, \ell_1 \in c_1, \ell_2 \in c_2, \ell_1 \neq \neg\ell_2\}$. Now let $\mathcal{P} = \{\{v_{\ell c} \mid \ell \in c\} \mid c \in \varphi\}$. Note that $\mathcal{P}$ is a partition of $V(\Gamma(\varphi))$ and every set in $\mathcal{P}$ corresponds to a clause in φ. Because all the clauses in φ are mutually satisfiable, the quotient $\Gamma(\varphi)/\mathcal{P}$ is a clique. Now we prove there is a model for φ iff there is a transversal of $\mathcal{P}$ consistent with $\Gamma(\varphi)$. ($\Rightarrow$) A model $\mathcal{M}$ has the property $\forall c \in \varphi.\ \exists \ell \in c.\ \mathcal{M} \models \ell$. Due to the correspondence between sets in $\mathcal{P}$ and clauses in φ, a transversal $\pitchfork$ of $\mathcal{P}$ can be constructed by selecting one satisfied literal from each clause. Based on a model, $\pitchfork$ will never select conflicting literals. All members of $\mathcal{R}(\pitchfork)$ are pairwise connected, so $\pitchfork$ is consistent with $\Gamma(\varphi)$ as required. ($\Leftarrow$) By definition $\pitchfork$ is consistent with $\Gamma(\varphi)$, so the subgraph of Γ induced by $\mathcal{R}(\pitchfork)$ is a clique. Let $\mathcal{L}$ be literals corresponding to $\mathcal{R}(\pitchfork)$, and note that its elements are pairwise consistent. A model $\mathcal{M}$ for φ is constructed by assigning v to $\top$ where v occurs positively in $\mathcal{L}$, and to $\bot$ otherwise. □

Theorem 2. *Problem 2 is NP-complete.*

Proof. Consistency of a transversal $\pitchfork$ is clearly polynomial-time testable, as is the coverage condition that $v \in \mathcal{R}(\pitchfork)$. Thus, we have membership in NP. NP-hardness follows from the following reduction from Problem 3 (P3). Taking $\mathcal{P}$ and Γ as inputs to P3, construct the graph Γ', where: $V(\Gamma') = V(\Gamma) \cup \{v\}$, v an auxiliary vertex; and $E(\Gamma') = E(\Gamma) \cup \{\{v, u\} \mid u \in V(\Gamma)\}$. There is a solution to P3 with inputs Γ and $\mathcal{P}$ iff P2 is soluble with inputs Γ', $\mathcal{P} \cup \{\{v\}\}$ (= $\mathcal{P}'$) and v. ($\Rightarrow$) If $\pitchfork$ is a transversal of $\mathcal{P}$ consistent with Γ, then $\pitchfork \cup \{\{v\} \mapsto v\} (= \pitchfork')$ is a transversal of $\mathcal{P}'$. As v is fully connected, $\pitchfork'$ is consistent with Γ'. $\pitchfork'$ is a solution to P2 because $v \in \mathcal{R}(\pitchfork')$. ($\Leftarrow$) If $\pitchfork'$ is a solution to P2 with inputs Γ', $\mathcal{P}'$ and v, then $\pitchfork'$ is a transversal of $\mathcal{P}'$, and also $\mathcal{P}$. All edges in $E(\Gamma')$ are in $E(\Gamma)$, with the exception of those adjacent v, and since $\pitchfork'$ is consistent with Γ', then $\pitchfork'$ is consistent with Γ. Thus, $\pitchfork'$ solves P3. □

Note: the NP-hard canonicalisation problem—the optimisation problem posed at each state encountered by orbit search—is not known to be in NP. Our above results imply that exploitation of symmetry via instantiation poses a decision problem in NP that needs to be solved only once.

5 Concrete Plan from Quotient Plan

Having computed isomorphic subproblems Δ that cover the goals of Π, a concrete plan is a concatenation of plans for members of Δ. However, this is not always straightforward.

Example 5. *Take Π_2, $\mathcal{P}$ and $\pitchfork$ from Example 2. Note $\pitchfork' = \{p_1 \mapsto x, p_2 \mapsto b, p_3 \mapsto n\}$ is also an instantiation of $\Pi_2/\mathcal{P}$ consistent with Π_2. Observe that $\{\pitchfork(\!|\Pi_2/\mathcal{P}|\!), \pitchfork'(\!|\Pi_2/\mathcal{P}|\!)\}$ covers the concrete goal $G = \{\overline{m}, \overline{n}\}$. A plan for $\Pi_2/\mathcal{P}$ is $\dot\pi' = (\{p_1, p_2\}, \{\overline{p}_3, \overline{p}_1\})$ and its two instantiations are $\pitchfork(\!|\dot\pi'|\!) = (\{x, a\}, \{\overline{m}, \overline{x}\})$ and $\pitchfork'(\!|\dot\pi'|\!) = (\{x, b\}, \{\overline{n}, \overline{x}\})$. Concatenating $\pitchfork(\!|\dot\pi'|\!)$ and $\pitchfork'(\!|\dot\pi'|\!)$ in any order does not solve Π_2 because both plans require x initially, but do not establish it. To overcome this issue in practice, before we solve $\Pi_2/\mathcal{P}$ we augment its goal with the assignment $p_1 \mapsto \top$.*

We now give conditions under which concatenation is valid, and detail the quotient-goal augmentation step we use to ensure validity in practice. We shall use the well-known notion of *projection*, where $Y\restriction_X$ is a version of Y with all elements mentioning an element not in X removed.[1] In addition, we will write $Y\restriction^X$ to mean $Y\restriction_{\mathcal{D}(X)}$, for the common case where we wish to project with respect to all the variables in the domain of a problem, state or set of assignments.

Definition 11 (Needed Assignments, Preceding Problem). *Needed assignments, $\mathcal{N}(\Pi)$, are assignments in the preconditions of actions and goal conditions that also occur in I, i.e., $\mathcal{N}(\Pi) = (\mathsf{pre}(A) \cup G) \cap I$. Problem Π_1 is said to precede Π_2, written $\Pi_1 \rhd \Pi_2$, iff*

$$G_1\restriction^{\mathcal{N}(\Pi_2)} = (I_2\restriction^{\Pi_1})\restriction^{\mathcal{N}(\Pi_2)} \ \wedge\ G_1\restriction^{\Pi_2} = G_2\restriction^{\Pi_1}$$

(i) The needed assignments of Π_2 which a plan for Π_1 could modify occur in G_1, and

(ii) G_2 contains all the assignments in G_1 which a plan for Π_2 might modify.

Example 6. *Consider Π_2 and Π_3 from Examples 2 and 3 respectively. $\mathcal{N}(\Pi_2) = \{x, a, b\}$ and $\mathcal{N}(\Pi_3) = G_1 = \{\overline{m}, \overline{n}\}$. Since $G_2\restriction^{\mathcal{N}(\Pi_3)} = G_2$, $(I_3\restriction^{\Pi_2})\restriction^{\mathcal{N}(\Pi_3)} = G_2$, $G_2\restriction^{\Pi_3} = G_2$ and $G_3\restriction^{\Pi_2} = G_2$, we have $\Pi_2 \rhd \Pi_3$.*

Writing $\dot\pi_i{\cdot}\dot\pi_j$ for concatenation of plans $\dot\pi_i, \dot\pi_{i+1} \ldots \dot\pi_j$, a simple inductive argument gives the following:

Lemma 2. *Let $\Pi_1 \ldots \Pi_N$ be a set of problems satisfying $\Pi_j \rhd \Pi_k$ for all $j < k \leq N$. For $1 \leq i \leq N$ if state s satisfies $I_i \subseteq s$ and $\dot\pi_i$ solves Π_i, then $\mathsf{e}(s, \dot\pi_1{\cdot}\dot\pi_N)\restriction^{G_i} = G_i$.*

Take problem Π and a set of problems $\mathbf{\Pi}$, we say that $\mathbf{\Pi}$ covers Π iff $\forall g \in G.\ \exists \Pi' \in \mathbf{\Pi}.\ g \in G'$ and $\forall \Pi' \in \mathbf{\Pi}.\ \Pi' \subseteq \Pi$. *I.e.*, every goal from Π is stated in one or more members of $\mathbf{\Pi}$, a set of subproblems of Π.

Theorem 3. *Consider a set $\Pi_1 \ldots \Pi_N$ of problems that covers Π, satisfying $\Pi_j \rhd \Pi_k$ for all $j < k \leq N$. For $1 \leq i \leq N$ if $\dot\pi_i$ is a plan for Π_i, then $\dot\pi_1{\cdot}\dot\pi_N$ is a plan for Π.*

This theorem follows directly from the fact that the set of problems covers Π and from Lemma 2.

We now address the question: When can plans for a set of isomorphic subproblems be concatenated to provide a concrete plan? We provide sufficient conditions in terms of the concepts of *common* and *sustainable* variables.

[1] A formal definition was given recently by Helmert *et al.* (2014).

Definition 12 (Common Variables). *For a set of instantiations Δ, the set of* common variables*, written $\bigcap_v \Delta$, comprises variables in $\bigcup_{\pitchfork \in \Delta} \mathcal{R}(\pitchfork)$ that occur in the ranges of more than one member of Δ.*

Definition 13 (Sustainable Variables). *A set of variables V in a problem Π is sustainable iff $I\downarrow_V = G\downarrow_V$.*

Theorem 4. *Take problem Π, partition $\mathcal{P}$ of $\mathcal{D}(\Pi)$ where the quotient $\Pi/\mathcal{P}(= \Pi')$ is well-defined, with solution $\dot{\pi}'$, and consistent instantiations Δ. Suppose $\{\pitchfork(\!|\Pi'|\!) \mid \pitchfork \in \Delta\}(= \mathbf{\Pi})$ covers Π, and $\mathcal{Q}(\!|\bigcap_v \Delta|\!) \cap \mathcal{D}(\mathcal{N}(\Pi'))$—i.e., based on Definition 7, the orbits of common variables from needed assignments—are sustainable in Π'. Then any concatenation of the plans $\{\pitchfork(\!|\dot{\pi}'|\!) \mid \pitchfork \in \Delta\}$ solves Π.*

Proof. Identify the elements in Δ by indices in $\{1..|\Delta|\}$. Let $k \in \{1..|\Delta|\}$ and $\Pi_k = \pitchfork_k(\!|\Pi'|\!)$, and note $\mathcal{R}(\pitchfork_k) = \mathcal{D}(\Pi_k)$. Take $\pitchfork_i, \pitchfork_j \in \Delta$, where $i, j \in \{1..|\Delta|\}$ and $i \neq j$. We first show that $\Pi_i \rhd \Pi_j$. For any $p \in \mathcal{P}$, $\pitchfork_i(p) = \pitchfork_j(p)$ if $\pitchfork_i(p) \in \mathcal{R}(\pitchfork_i) \cap \mathcal{R}(\pitchfork_j)$. Therefore, $I_i\downarrow^{\Pi_j} = I_j\downarrow^{\Pi_i}$ and $G_i\downarrow^{\Pi_j} = G_j\downarrow^{\Pi_i}$, providing the right conjunct in Definition 11. For the left conjunct, note $\mathcal{D}(\mathcal{N}(\Pi_j)) \subseteq \mathcal{D}(\Pi_j)$ and $\mathcal{D}(\Pi_i) \cap \mathcal{D}(\Pi_j) \subseteq \bigcap_v \Delta$. The sustainability premise—$\mathcal{Q}(\bigcap_v \Delta) \cap \mathcal{D}(\mathcal{N}(\Pi'))$ is sustainable in Π'—then provides that $\mathcal{D}(\Pi_i) \cap \mathcal{D}(\mathcal{N}(\Pi_j))$ is sustainable in Π_i—*i.e.*, $I_i\downarrow^{\mathcal{N}(\Pi_j)} = G_i\downarrow^{\mathcal{N}(\Pi_j)}$. Thus $G_i\downarrow^{\mathcal{N}(\Pi_j)} = (I_j\downarrow^{\Pi_i})\downarrow^{\mathcal{N}(\Pi_j)}$, and we conclude $\Pi_i \rhd \Pi_j$. Since a plan $\pitchfork_k(\!|\dot{\pi}'|\!)$ solves $\pitchfork_k(\!|\Pi'|\!)$, $(\pitchfork_k(\!|\Pi'|\!)).I \subseteq \Pi.I$, therefore as per Theorem 3 a solution to Π is $\dot{\pi}_1 \cdot \dot{\pi}_N$. □

In practice we take $V^* = \mathcal{Q}(\!|\bigcap_v \Delta|\!) \cap \mathcal{D}(\mathcal{N}(\Pi/\mathcal{P}))$, and augment the goal of the quotient $\Pi/\mathcal{P}$ by adding $(\Pi/\mathcal{P}).I\downarrow^{V^*}$. Call the resulting problem Π^q and its solution $\dot{\pi}^q$. Theorem 4 shows that any concatenation of the plans $\{\pitchfork(\!|\dot{\pi}^q|\!) \mid \pitchfork \in \Delta\}$ solves Π. Thus, our approach is sound.

Example 7. *Take Π_2, $\mathcal{P}$, $\pitchfork$ and $\pitchfork'$ from Example 5. There is one common variable, $\{x\} = \bigcap_v \Delta$ from the orbit $\{p_1\} = \mathcal{Q}(\!|\bigcap_v \Delta|\!)$. Here $\mathcal{N}(\Pi_2/\mathcal{P}) = \{p_1, p_2\}$, and the orbit of the common needed variable is $\{p_1\} = \mathcal{Q}(\!|\bigcap_v \Delta|\!) \cap \mathcal{N}(\Pi_2/\mathcal{P})$. To solve Π_2 via solving $\Pi_2/\mathcal{P}$, we augment the goal $(\Pi_2/\mathcal{P}).G$ with the assignment to p_1 in $(\Pi_2/\mathcal{P}).I$. The resulting problem, Π_2^q, is equal to $\Pi_2/\mathcal{P}$ except that it has the goals $\Pi_2^q.G = \{p_1, \overline{p}_3\}$. A plan for Π_2^q is $\dot{\pi}^q = (\{p_1, p_2\}, \{\overline{p}_3, \overline{p}_1\})(\{\}, \{p_1\})$, and two instantiations of it are $\pitchfork(\!|\dot{\pi}^q|\!) = (\{x, a\}, \{\overline{m}, \overline{x}\})(\{\}, \{x\})$ and $\pitchfork'(\!|\dot{\pi}^q|\!) = (\{x, b\}, \{\overline{n}, \overline{x}\})(\{\}, \{x\})$. Concatenating $\pitchfork(\!|\dot{\pi}^q|\!)$ and $\pitchfork'(\!|\dot{\pi}^q|\!)$ in any order solves Π_2.*

6 Experimental Results

Implemented in C++,[2] our approach uses: NAUTY\TRACES to calculate symmetries [McKay and Piperno, 2014]; Z3 to find isomorphic subproblems [de Moura and Bjørner, 2008]; and the initial-plan search by LAMA as the *base planner* [Richter and Westphal, 2010]. We limit *base planner* runtimes to 30 minutes.

[2] bitbucket.org/MohammadAbdulaziz/planning.git in codeBase.

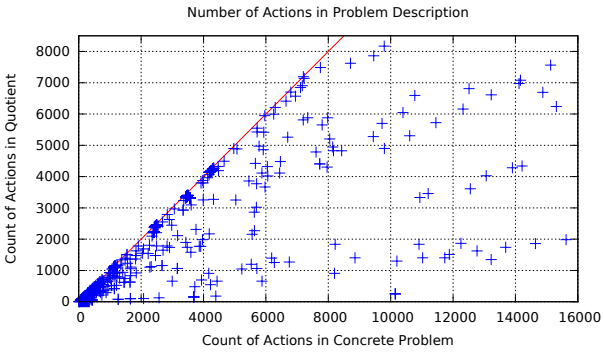


Figure 1: Scatter-plot comparing the number of *actions* in problem posed by descriptive quotient *vs.* concrete problem, with red line plotting $f(x) = x$.

By running our algorithm, we obtained a set of benchmarks with soluble descriptive quotients whose solutions can be instantiated to concrete plans. That set includes 439 problems, from 16 IPC benchmarks and 4 benchmarks from [Porco *et al.*, 2013]. In all our experimentation we identified 120 instances where we were able to confirm that the descriptive quotient does not have a solution and the concrete problem does. Figure 1 plots the sizes, in terms of the number of actions, of concrete and quotient problem descriptions. The plotted data shows that the descriptive quotient can be much smaller than its concrete counterpart. In just over 15% of instances the quotient has less than half the number of actions. Here, we also analyzed what aspects of our approach are computationally expensive. In 79% of cases 99% of the runtime of our approach is executing the base planner. In 96% of cases 95% of the runtime of our approach is executing the base planner. Overall, 3% of time is spent in instantiation and finding chromatic numbers.

We examined where our approach is comparatively scalable and fast compared to the base planner. For 430 instances where the base planner and our approach are successful, Figure 2 displays the speedup factors where planning *via* the descriptive quotient is comparatively fast. Overall, for 68% of instances our approach is comparatively fast, and in 15% planning *via* the quotient is at least twice as fast. With few exceptions, instances where our approach is at least twice as fast are from GRIPPER, HIKING, MPRIME, MYSTERY, PARCPRINTER, PIPESWORLD, TPP and VISITALL. In 5 problems from HIKING, 2 from PARCPRINTER, 1 from TETRIS and 1 from KCOLOURABILITY, the base planner cannot solve the concrete problem, but can solve the quotient. Planning *via* the quotient can also be slow, primarily due to the extra cost of finding symmetries,[3] and because LAMA is heuristic and occasionally finds that the quotient poses a more challenging problem. Figure 3 provides the dual to Figure 2, showing cases where planning directly for the concrete problem

[3] If symmetries are given as part of the problem description, or if one resorts to more heuristic methods of discovering them, such as those described by [Guere and Alami, 2001; Fox and Long, 2002], this burden is relieved.

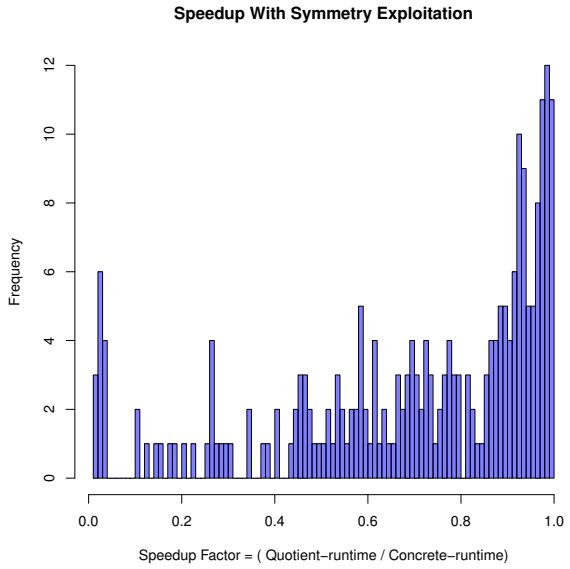


Figure 2: Plots histogram of speedup factors experienced when planning **with** symmetry exploitation, reporting only for instances where planning via the quotient is faster, and only for instances where the base planner takes > 5 seconds.

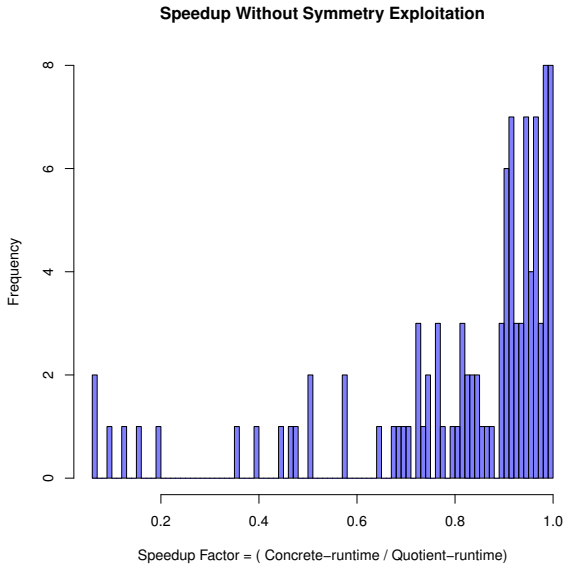


Figure 3: Plots histogram of speedup factors experienced when planning **without** symmetry exploitation, reporting only for instances where planning for the concrete problem directly is faster, and only for instances where the base planner takes > 5 seconds.

is comparatively fast. This is the case in 32% instances, and indeed in 2% our approach is at least twice as slow.

Finally, it is worth highlighting the difference between searching for a plan in the state-space of the descriptive quotient versus approximate orbit search—*i.e.*, searching for a plan in an approximation of the quotient state-space—as is done in the state-of-the-art techniques for exploiting symmetries in planning.[4] In comparing those approaches, we measure the number of states expanded using a breadth-first search. We consider the IPC instances GRIPPER-20 and MPRIME-21, the largest instances from domains GRIPPER and MPRIME reported solved by Pochter *et al.* (2011) using breadth-first search. Those authors report the number of states expanded to be 60.5K and 438K, respectively. Using that search to solve the problem posed by our descriptive quotient, we expand 6 and 203K states, respectively.

7 Conclusions, Related and Future Work

We plan with symmetries using the small *descriptive quotient* of the concrete problem description. A concrete plan is obtained by concatenating instantiations of the plan for the *descriptive quotient*. Unlike existing approaches, our search for a plan does not need to reason about symmetries between concrete states and the effects of actions on those. Plan search can be performed by an off-the-shelf SAT/CSP solver, in which case *symmetry breaking* constraints are not required. Alternatively, using a state-based planner we avoid repeated (approximate) solution to the intractable canonicalisation problem, a clear bottleneck of recent planning algorithms. In this respect, our approach is similar to searching in a *counting abstraction*, as surveyed by [Wahl and Donaldson, 2010]. That approach treats a *transition system* isomorphic to the *quotient transition system*. That system has a state-space which can be exponentially larger than that of a descriptive quotient—*i.e.*, the descriptive quotient will model 1 object, whereas the quotient *transition system* models N symmetric objects. Existing state-based methods also plan in that relatively large *quotient system*, and face an additional intractable problem for every encountered state. By employing approximate canonicalisation, such methods face a state-space much larger than that posed by the quotient system, which can be exponentially larger than that posed by a descriptive quotient.

Our approach decomposes a problem into subproblems, and in that respect is related to factored planning [Amir and Engelhardt, 2003; Brafman and Domshlak, 2006; Kelareva *et al.*, 2007]. There are close ties between factored planning, identifying tractable classes of planning problem, and derivation of tight plan length *upper bounds*—*i.e.*, in the sense developed in [Rintanen and Gretton, 2013]. We therefore note that (i) the relationship between symmetry and upper bounds has been explored previously, and (ii) symmetries have started to be explored in a factored planning setting. Guere and Alami (2001) propose to plan via a *shape-graph*, a compact description of the problem state space in which states are represented by equivalence classes of symmetric states. As well as planning by searching in that graph, using the *diameter* of the *shape-graph* Guere and Alami are able to calculate tight upper bounds for the highly symmetric GRIPPER and BLOCKS-WORLD domains. Symme-

[4] Such a comparison is admittedly unfair, as the problem posed by a descriptive quotient is not a bisimulation of the concrete problem.

tries were explored in factored planning in the context of *merge-and-shrink* heuristics [Helmert *et al.*, 2014]. Sievers *et al.* (2015) developed a symmetry guided *merging* operation which yields relatively compact heuristic models, improving the scalability of that approach. Future research may explore descriptive quotients: to develop heuristics, to characterise tractable classes, and to develop bounding methods.

Acknowledgement

We would like to thank Brendan McKay, Miquel Ramírez, Patrik Haslum and Alban Grastien for useful discussions.

References

[Amir and Engelhardt, 2003] Eyal Amir and Barbara Engelhardt. Factored planning. In *IJCAI*, volume 3, pages 929–935. Citeseer, 2003.

[Brafman and Domshlak, 2006] Ronen I Brafman and Carmel Domshlak. Factored planning: How, when, and when not. In *AAAI*, volume 6, pages 809–814, 2006.

[Brown *et al.*, 1996] Cynthia A. Brown, Larry Finkelstein, and Paul Walton Purdom Jr. Backtrack searching in the presence of symmetry. *Nord. J. Comput.*, 3(3):203–219, 1996.

[Clarke *et al.*, 1998] Edmund M Clarke, E Allen Emerson, Somesh Jha, and A Prasad Sistla. Symmetry reductions in model checking. In *Computer Aided Verification*, pages 147–158. Springer, 1998.

[Crawford *et al.*, 1996] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. *KR*, 96:148–159, 1996.

[de Moura and Bjørner, 2008] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver, 2008.

[Domshlak *et al.*, 2012] Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Enhanced symmetry breaking in cost-optimal planning as forward search. In *ICAPS*, 2012.

[Domshlak *et al.*, 2013] Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Symmetry breaking: Satisficing planning and landmark heuristics. In *ICAPS*, 2013.

[Eugene, 1993] M Eugene. Permutation groups and polynomial-time computation. In *Groups and Computation: Workshop on Groups and Computation, October 7-10, 1991*, volume 11, page 139. American Mathematical Soc., 1993.

[Fox and Long, 1999] Maria Fox and Derek Long. The detection and exploitation of symmetry in planning problems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 956–961, 1999.

[Fox and Long, 2002] Maria Fox and Derek Long. Extending the exploitation of symmetries in planning. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, pages 83–91, 2002.

[Guere and Alami, 2001] Emmanuel Guere and Rachid Alami. One action is enough to plan. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 439–444, 2001.

[Helmert *et al.*, 2014] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM (JACM)*, 61(3):16, 2014.

[Joslin and Roy, 1997] David Joslin and Amitabha Roy. Exploiting symmetry in lifted CSPs. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island.*, pages 197–202, 1997.

[Kelareva *et al.*, 2007] Elena Kelareva, Olivier Buffet, Jinbo Huang, and Sylvie Thiébaux. Factored planning using decomposition trees. In *IJCAI*, pages 1942–1947, 2007.

[McKay and Piperno, 2014] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.

[Miguel, 2001] Ian Miguel. Symmetry-breaking in planning: Schematic constraints. In *Proceedings of the CP01 Workshop on Symmetry in Constraints*, pages 17–24, 2001.

[Pochter *et al.*, 2011] Nir Pochter, Aviv Zohar, and Jeffrey S Rosenschein. Exploiting problem symmetries in state-based planners. In *AAAI*, 2011.

[Porco *et al.*, 2013] Aldo Porco, Alejandro Machado, and Blai Bonet. Automatic reductions from PH into STRIPS or how to generate short problems with very long solutions. In *ICAPS*, 2013.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[Rintanen and Gretton, 2013] Jussi Rintanen and Charles Gretton. Computing upper bounds on lengths of transition sequences. In *IJCAI*, 2013.

[Rintanen, 2003] Jussi Rintanen. Symmetry reduction for SAT representations of transition systems. In *ICAPS*, pages 32–41, 2003.

[Shleyfman *et al.*, 2015] Alexander Shleyfman, Michael Katz, Malte Helmert, Silvan Sievers, and Martin Wehrle. Heuristics and symmetries in classical planning. In *AAAI*, 2015.

[Sievers *et al.*, 2015] Silvan Sievers, Martin Wehrle, Malte Helmert, Alexander Shleyfman, and Michael Katz. Factored symmetries for merge-and-shrink abstractions. In *Proc. 29th National Conf. on Artificial Intelligence*. AAAI Press, 2015.

[Wahl and Donaldson, 2010] Thomas Wahl and Alastair F. Donaldson. Replication and abstraction: Symmetry in automated formal verification. *Symmetry*, 2(2):799–847, 2010.