Operator Mutexes and Symmetries for Simplifying Planning Tasks

Daniel Fišer
Czech Technical University in Prague,
Faculty of Electrical Engineering,
Prague, Czech Republic
danfis@danfis.cz

Álvaro Torralba
Saarland University,
Saarland Informatics Campus,
Saarbrücken, Germany
torralba@cs.uni-saarland.de

Alexander Shleyfman
Technion
Haifa, Israel
shleyfman.alexander@gmail.com

Abstract

Simplifying classical planning tasks by removing operators while preserving at least one optimal solution can significantly enhance the performance of planners. In this paper, we introduce the notion of operator mutex, which is a set of operators that cannot all be part of the same (strongly) optimal plan. We propose four different methods for inference of operator mutexes and experimentally verify that they can be found in a sizable number of planning tasks. We show how operator mutexes can be used in combination with structural symmetries to safely remove operators from the planning task.

Introduction

The solution to a classical planning problem is a sequence of operators leading from the initial state to one of the goal states. Since classical planning problems are described in a domain-independent manner, automatic extraction of structural information can provide useful guidance for planners.

In this paper, we focus on identifying mutually exclusive operators, in the sense that applying one forbids to apply the others in the shortest optimal plan. For example, in a task with non-replenishable resources, at most one operator depleting a resource can be applied in any optimal plan. In general, we say that a set of operators is a strong operator mutex (op-mutex) if no strongly optimal plan contains all of them. We introduce several methods to infer op-mutexes automatically from the description of the task, and show that they can be found in a sizable number of domains.

We combine op-mutexes with structural symmetries in order to identify operators that can be safely removed from the planning tasks. The notion of symmetries is not new to classical planning (Fox and Long 1999). They can be used to reduce the size of the search space in heuristic search planners (Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012), obtain more compact encodings in SAT-based planners (Rintanen 2003), compute better heuristics (Domshlak, Katz, and Shleyfman 2013; Sievers et al. 2015; 2017), transform the task (Riddle et al. 2015), or ground the task (Röger, Sievers, and Katz 2018). Here, we use symmetries to prove that certain operators can be removed from the planning task. We show that at least one optimal plan is preserved when removing some operators if they are all op-mutex with other symmetric operators. In order to find out whether more operators can be removed, we then analyze which symmetries are preserved after removing a set of operators in this way. This leads to a fixpoint computation method that can remove a sizable number of operators in some domains.

Background

A STRIPS **planning task** Π is specified by a tuple $\Pi = \langle \mathcal{F}, \mathcal{O}, I, G \rangle$, where $\mathcal{F} = \{f_1, \ldots, f_n\}$ is a set of facts, and $\mathcal{O} = \{o_1, \ldots, o_m\}$ is a set of grounded operators. A **state** $s \subseteq \mathcal{F}$ is a set of facts, $I \subseteq \mathcal{F}$ is an **initial state** and $G \subseteq \mathcal{F}$ is a **goal** specification. An **operator** o is a tuple $o = \langle \text{pre}(o), \text{add}(o), \text{del}(o), \text{c}(o) \rangle$, where $\text{pre}(o) \subseteq \mathcal{F}$ is a set of preconditions of the operator o, and $\text{add}(o) \subseteq \mathcal{F}$ and $\text{del}(o) \subseteq \mathcal{F}$ are sets of add and delete effects, respectively, and $\text{c}(o) \in \mathbb{R}_0^+$ is a cost of the operator. All operators are well-formed, i.e., $\text{add}(o) \cap \text{del}(o) = \emptyset$ and $\text{pre}(o) \cap \text{add}(o) = \emptyset$. An operator o is **applicable** in a state s if $\text{pre}(o) \subseteq s$. The **resulting state** of applying an applicable operator o in a state s is the state $o[s] = (s \setminus \text{del}(o)) \cup \text{add}(o)$. A state s is a **goal state** iff $G \subseteq s$.

A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is applicable in a state s_0 if there are states $s_1, \ldots, s_n$ such that o_i is applicable in s_{i-1} and $s_i = o_i[s_{i-1}]$ for $i \in \{1, \ldots, n\}$. The resulting state of this application is $\pi[s_0] = s_n$ and $\text{c}(\pi) = \sum_{o \in \pi} \text{c}(o)$ denotes the cost of this sequence of operators. By $|\pi|$ we denote the length of the sequence. π is called a **plan** iff $\pi[I] \supseteq G$, π is called an **optimal plan** if its cost is minimal among all plans, and π is called a **strongly optimal plan** if it is an optimal plan and it contains the minimum number of operators among optimal plans. We denote the set of strongly optimal plans by $\mathcal{P}_\Pi$.

A state s is **reachable** if there exists an applicable operator sequence π such that $\pi[I] = s$. The set of all reachable states is denoted by $\mathcal{R}_\Pi$. An operator o is **reachable** iff it is applicable in some reachable state. A state s is a **dead-end state** iff $G \not\subseteq s$ and there is no applicable operator sequence π such that $G \subseteq \pi[s]$. A **mutex** $M \subseteq \mathcal{F}$ is a set of facts such that for every reachable state $s \in \mathcal{R}_\Pi$ it holds that $M \not\subseteq s$. A **mutex group** $M \subseteq \mathcal{F}$ is a set of facts such that $|M \cap s| \leq 1$ for every reachable state $s \in \mathcal{R}_\Pi$.

A **labeled transition system** (LTS) is a tuple $\Theta = \langle \mathcal{S}, L, T, s_I, S_\star \rangle$, where $\mathcal{S}$ is a finite set of **states**, L is a finite set of **labels** with associated cost $\mathrm{c}(l) \in \mathbb{R}_0^+$ to each label $l \in L$, $T \subseteq \mathcal{S} \times L \times \mathcal{S}$ is a set of **transitions**, $s_I \in \mathcal{S}$ is the **initial state**, and $S_\star \subseteq \mathcal{S}$ is a set of **goal states**. We write $s_1 \xrightarrow{l} s_2$ to refer to a transition from s_1 to s_2 with the label l. A sequence of labels $\langle l_1, \ldots, l_n \rangle$ is a **path** from s_0 to s_n in Θ if there exist $s_{i-1} \xrightarrow{l_i} s_i \in T$ for every $i \in \{1, \ldots, n\}$. We say that s' is **reachable from** s if there is a path from s to s'.

The **state space** of a planning task Π is the LTS Θ_Π where $\mathcal{S} := \mathcal{R}_\Pi$, $s_I := I$, $s \in S_\star$ iff $G \subseteq s$, the labels L are the operators $\mathcal{O}$ with the given costs, and $s \xrightarrow{o} s'$ is a transition in T if $\mathrm{pre}(o) \subseteq s$ and $o[s] = s'$.

An **abstraction** α for a transition system Θ is a function mapping states $\mathcal{S}$ into a set of abstract states $\mathcal{S}^\alpha$. The **abstract transition system** Θ^α is defined as $\langle \mathcal{S}^\alpha, L, T^\alpha, s_I^\alpha, S_\star^\alpha \rangle$, where $\alpha(s) \xrightarrow{o} \alpha(s') \in T^\alpha$ iff $s \xrightarrow{o} s' \in T$, $s_I^\alpha = \alpha(s_I)$, and $S_\star^\alpha = \{\alpha(s) \mid s \in S_\star\}$.

A **projection** of the state space Θ_Π to the set of facts $F \subseteq \mathcal{F}$ is an abstract transition system $\Theta_\Pi^{\alpha_F}$ with the abstraction $\alpha_F(s) = s \cap F$.

For notational convenience, we will sometimes abuse the notation for sequences by referring to a sequence as a set. For a set of operators $O \subseteq \mathcal{O}$, we denote $\Pi \setminus O$ the planning task resulting from removing operators O, $\Pi \setminus O = \langle \mathcal{F}, \mathcal{O} \setminus O, I, G \rangle$.

Operator Mutexes and Redundancy

Our goal in this paper is to eliminate operators from a planning task, if they can be proven unnecessary to find a strongly optimal plan. We say that a set of operators is redundant if removing them from the task preserves at least one strongly optimal plan.

Definition 1. Given the planning task Π, a set of operators $O \subseteq \mathcal{O}$ is **redundant** if $\mathcal{P}_\Pi \neq \emptyset$ implies $\mathcal{P}_{\Pi \setminus O} \cap \mathcal{P}_\Pi \neq \emptyset$.

The union of two redundant sets is not necessarily also a redundant set. However, we can merge two sets of redundant operators, if one is shown to be redundant in the task where the other has already been removed.

Proposition 2. *Let Π denote a planning task with a set of operators $\mathcal{O}$ and let $R_1, R_2 \subseteq \mathcal{O}$, $R_1 \cap R_2 = \emptyset$. If R_1 is redundant in Π and R_2 is redundant in $\Pi \setminus R_1$, then $R_1 \cup R_2$ is redundant in Π.*

Proof. $\mathcal{P}_\Pi \neq \emptyset$ implies $\mathcal{P}_{\Pi \setminus R_1} \cap \mathcal{P}_\Pi \neq \emptyset$, because R_1 is redundant in Π. And $\mathcal{P}_{\Pi \setminus R_1} \neq \emptyset$ implies $\mathcal{P}_{\Pi \setminus (R_1 \cup R_2)} \cap \mathcal{P}_{\Pi \setminus R_1} \neq \emptyset$, because R_1 is redundant in $\Pi \setminus R_1$. Therefore $\mathcal{P}_{\Pi \setminus (R_1 \cup R_2)} \subseteq \mathcal{P}_{\Pi \setminus R_1} \subseteq \mathcal{P}_\Pi$ and $\mathcal{P}_{\Pi \setminus (R_1 \cup R_2)} \neq \emptyset$. □

The notion of mutually exclusive facts that cannot be together in any reachable state has proven to be very useful to improve different types of planning algorithms. Here, we define strong operator mutexes as sets of operators that cannot be part of the same strongly optimal plan.

Definition 3. A **strong operator mutex** (op-mutex) $O \subseteq \mathcal{O}$ is a nonempty set of operators s.t. $O \not\subseteq \pi$ for every $\pi \in \mathcal{P}_\Pi$.

The mutual exclusion between operators with respect to strongly optimal plans means that every op-mutex always contains at least one operator that is redundant.

Proposition 4. *In every op-mutex $O \subseteq \mathcal{O}$, there is an operator $o \in O$ such that $\{o\}$ is redundant.*

Proof. Given $\pi \in \mathcal{P}_\Pi$, $O \not\subseteq \pi$ by Definition 3, thus there is an operator $o \in O$ s.t. $o \notin \pi$, therefore $\pi \in \mathcal{P}_{\Pi \setminus \{o\}}$. □

From now on, we will concentrate on the special case of op-mutexes that are formed by pairs of operators, since, as shown by Proposition 5, they can be used to identify redundant sets consisting of more than one operator.

Proposition 5. *Let Π denote a planning task, let $O_1, O_2 \subseteq \mathcal{O}$ s.t. $O_1 \cap O_2 = \emptyset$ and $\{o_1, o_2\}$ is an op-mutex for every $o_1 \in O_1$ and every $o_2 \in O_2$, then O_1 or O_2 is redundant.*

Proof Sketch. If $o \notin \pi$ for every $o \in O_1$ and every $\pi \in \mathcal{P}_\Pi$, then $\mathcal{P}_{\Pi \setminus O_1} = \mathcal{P}_\Pi$. Otherwise there exists $o \in O_1$ s.t. $o \in \pi$, for some $\pi \in \mathcal{P}_\Pi$, and since $\{o, o'\}$ is an op-mutex for each $o' \in O_2$, $o' \notin \pi$. Thus, O_2 is redundant. □

Op-mutexes are useful for obtaining candidate sets of operators that may be redundant. However, op-mutexes are not sufficient to prove any such set redundant. Later, we will show how to combine op-mutexes with symmetries to identify which set of operators is actually redundant.

Inference of Operator Mutexes

In this section, we describe several methods for inference of op-mutexes based on well-known planning techniques.

Abstractions

Abstraction heuristics map the state space of a planning task into a smaller abstract state space and use the distance in the abstract state space as an admissible estimation. Abstractions can also be used to infer op-mutexes.

Theorem 6. *Let Π denote a planning task, and let $o_1, o_2 \in \mathcal{O}$, $o_1 \neq o_2$, denote operators in Π. If there exists an abstract transition system Θ_Π^α such that for every transition $s_1 \xrightarrow{o_1} s_1'$ and every transition $s_2 \xrightarrow{o_2} s_2'$, s_2 is not reachable from s_1' and s_1 is not reachable from s_2', then $\{o_1, o_2\}$ is an op-mutex.*

Proof Sketch. The theorem clearly holds for Θ_Π, since o_2 is never reachable from o_1 and vice versa. This means that o_1 and o_2 are never part of the same path, and therefore they cannot coexist in the same plan.

In their work, Helmert, Haslum, and Hoffmann (2007) showed that abstractions preserve all paths in the state space, thus if o_1 and o_2 are not reachable from one another in Θ_Π^α, then the same holds for Θ_Π. □

Theorem 6 shows how we can infer strong operator mutexes using any known method for computing abstractions of planning tasks, including pattern databases (Culberson and Schaeffer 1996; Edelkamp 2001), merge-and-shrink (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2014), or Cartesian abstractions (Seipp and Helmert 2018). Analyzing

what abstraction methods are best suited to find op-mutexes is out of the scope of this work, so we will focus our evaluation only on projections to individual mutex groups.

To compute op-mutexes with this method one needs to check reachability between every pair of states in the abstract state space Θ^{α}_{Π}. However, this can be done more efficiently by considering instead a smaller abstract state space Θ^{γ}_{Π}, where $\gamma(s) = \gamma(s')$ if $\alpha(s)$ and $\alpha(s')$ belong to the same strongly connected component in Θ^{α}_{Π}. In other words, given an abstract LTS in which we want to look for op-mutexes, one can always apply a condensation of its strongly connected components. The set of inferred op-mutexes will not be affected because if $\alpha(s)$ and $\alpha(s')$ belong to the same strongly connected component, then they have the same set of reachable abstract states.

Operators-as-Facts Compilation

Mutually exclusive facts have been well studied in the planning literature and there is a number of methods to automatically infer them (Gerevini and Schubert 1998; Rintanen 2000; Haslum and Geffner 2000; Fišer and Komenda 2018). We devise a compilation that allows us to use methods for inferring mutexes to infer op-mutexes. The idea is to transform the task by adding one artificial fact per operator, so that two operators are op-mutex if their corresponding artificial facts are mutex.

Definition 7. Given the planning task $\Pi = \langle \mathcal{F}, \mathcal{O}, I, G \rangle$, the **op-fact compilation** of Π is another planning task $\Pi_{\text{op}} = \langle \mathcal{F} \cup \mathcal{F}_{\text{op}}, \mathcal{O}_{\text{op}}, I, G \rangle$, where $\mathcal{F}_{\text{op}} = \{f_o \mid o \in \mathcal{O}\}$, and $\mathcal{O}_{\text{op}} = \{o^\diamond \mid o \in \mathcal{O}\}$ where each operator $o^\diamond$ is defined as $o^\diamond = \langle \text{pre}(o), \text{add}(o) \cup \{f_o\}, \text{del}(o), \text{c}(o) \rangle$.

Theorem 8. *Let Π_{op} denote the op-fact compilation of the planning task Π, and let $F = \{f_{o_1}, \ldots, f_{o_n}\} \subseteq \mathcal{F}_{\text{op}}$. If F is a mutex in Π_{op}, then $\{o_1, \ldots, o_n\}$ is an op-mutex in Π.*

Proof Sketch. Every $f_o \in \mathcal{F}_{\text{op}}$ appears only in the add effect of the operator $o^\diamond$ and otherwise the operators $o^\diamond$ and o are identical. Therefore a sequence of operators $\pi^\diamond = \langle o^\diamond_1, \ldots, o^\diamond_n \rangle$ in Π_{op} is applicable in I iff the sequence $\pi = \langle o_1, \ldots, o_n \rangle$ of the original task Π is applicable in I. Moreover, it holds that $\pi^\diamond[I] = \pi[I] \cup \{f_o \mid o^\diamond \in \pi^\diamond\}$. Assume, for contradiction, that $F = \{f_{o_1}, \ldots, f_{o_n}\}$ is a mutex in Π_{op} and there is a path π in the task Π s.t. $\{o_1, \ldots, o_n\} \subseteq \pi$. This immediately leads to a contradiction, because $F \subseteq \pi^\diamond[I]$. □

The op-fact compilation allows us to infer op-mutexes by using known methods for inference of mutexes, including the well known h^m family of heuristics (Haslum and Geffner 2000). However, we should stress that not every inference method is suitable for this type of compilation. For example, the inference of fact-alternating mutex groups (Fišer and Komenda 2018) would fail to produce any mutex group (and thus any mutex) containing any fact from $\mathcal{F}_{\text{op}}$, because they appear only in add effects.

Critical-Path Heuristics

The alternative characterization of the h^m family of heuristics introduced by Haslum (2009) can also be adapted to compute op-mutexes directly, without considering artificial facts for every operator. Definition 9 differs from the definition provided by Haslum only in that the goal specification is empty, because we are only interested in computing the set of operators reachable from the initial state.

Definition 9. Let $\Pi = \langle \mathcal{F}, \mathcal{O}, I, G \rangle$ denote a planning task. The planning task $\Pi^m = \langle \mathcal{F}^m, \mathcal{O}^m, I^m, \emptyset \rangle$ consists of a set of facts $\mathcal{F}^m = \{\phi_c \mid c \subseteq \mathcal{F}, |c| \leq m\}$, a set of operators $\mathcal{O}^m$, an initial state $I^m = \{\phi_c \mid c \subseteq I, |c| \leq m\}$, and an empty goal specification. For each operator $o \in \mathcal{O}$ and for each subset of facts $F \subseteq \mathcal{F}$ such that $|F| < m$ and F is disjoint with $\text{add}(o)$ and $\text{del}(o)$, the planning task Π^m contains an operator $\omega_{o,F} \in \mathcal{O}^m$ with: $\text{pre}(\omega_{o,F}) = \{\phi_c \mid c \subseteq (\text{pre}(o) \cup F), |c| \leq m\}$, $\text{add}(\omega_{o,F}) = \{\phi_c \mid c \subseteq (\text{add}(o) \cup F), c \cap \text{add}(o) \neq \emptyset, |c| \leq m\}$, $\text{del}(\omega_{o,F}) = \emptyset$.

To compute op-mutexes, we consider the compiled planning task for every operator o, that approximates which operators are reachable from the result of applying o in any reachable state.

Definition 10. Given the planning task $\Pi = \langle \mathcal{F}, \mathcal{O}, I, G \rangle$, a set of mutexes $\mathcal{M} \subseteq 2^{\mathcal{F}}$ for Π, an operator $o \in \mathcal{O}$, and a natural number $m \geq 1$, the **m-$\mathcal{M}$-compilation of Π for o** is another planning task $\Pi^m_{\mathcal{M},o} = \langle \mathcal{F}^m, \mathcal{O}^m, I^m_{\mathcal{M},o}, \emptyset \rangle$, where $\mathcal{F}^m$ and $\mathcal{O}^m$ are the same as in Π^m, and $I^m_{\mathcal{M},o} = \{\phi_c \mid c \subseteq (o[\text{pre}(o)] \cup E_{\mathcal{M},o}), |c| \leq m, c \notin \mathcal{M}\}$, where $E_{\mathcal{M},o} = \{f \mid f \in (\mathcal{F} \setminus \text{del}(o)), (\{f\} \cup \text{pre}(o)) \notin \mathcal{M}^\star, (\{f\} \cup o[\text{pre}(o)]) \notin \mathcal{M}^\star\}$ and $\mathcal{M}^\star = \{M \cup N \mid M \in \mathcal{M}, N \subseteq \mathcal{F}\}$.

Note that $\mathcal{M}^\star$ is also a set of mutexes, because every superset of a mutex is also a mutex.

The m-$\mathcal{M}$-compilation for an operator o differs from Definition 9 only in the construction of the initial state. The initial state is constructed as an over-approximation of the union of all possible states resulting from the application of the operator o on some reachable state while taking into account given mutexes $\mathcal{M}$. So, for an empty $\mathcal{M}$, the initial state $I^m_{\mathcal{M},o}$ will contain all facts except those from the delete effect of o, because they can never be part of the state resulting from the application of o. But if we are given additional information in the form of mutexes, we can apply it to exclude from $I^m_{\mathcal{M},o}$ the facts that cannot be part of any state resulting from the application of o.

Theorem 11. *Let Π denote a planning task, $\mathcal{M} \subseteq 2^{\mathcal{F}}$ a set of mutexes for Π, and $o_1, o_2 \in \mathcal{O}$, $o_1 \neq o_2$, two reachable operators with corresponding m-$\mathcal{M}$-compilations $\Pi^m_{\mathcal{M},o_1}$ and $\Pi^m_{\mathcal{M},o_2}$. If $\omega_{o_2,\emptyset}$ is not reachable in $\Pi^m_{\mathcal{M},o_1}$ and $\omega_{o_1,\emptyset}$ is not reachable in $\Pi^m_{\mathcal{M},o_2}$, then $\{o_1, o_2\}$ is an op-mutex.*

Proof Sketch. Haslum (2009) showed that if $\omega_{o,\emptyset}$ is not reachable in Π^m, then o is not reachable in Π. So we need to show that for every reachable state $s \in \mathcal{R}_\Pi$ such that o_1 is applicable in s it holds that $\{\phi_c \mid c \subseteq o_1[s], |c| \leq m\} \subseteq I^m_{\mathcal{M},o_1}$, because $\Pi^m_{\mathcal{M},o_1}$ is a delete-free problem and if o_2 is not reachable from the superset of all possible reachable states resulting from the application of the operator o_1, then o_2 cannot be reachable after any application of o_1 in Π (and the same for the reachability of o_1 from o_2). The rest follows directly from Theorem 6 with the identity abstraction $\alpha(s) = s$.

Let $s \in \mathcal{R}_\Pi$ denote a reachable state such that $\text{pre}(o_1) \subseteq s$, and let $S = \{c \mid c \subseteq o_1[s], |c| \leq m\}$. Since s is reachable, $o_1[s]$ is reachable, therefore $S \cap \mathcal{M} = \emptyset$. Now it is enough to show that $o_1[s] \subseteq (o_1[\text{pre}(o_1)] \cup E_{\mathcal{M},o_1})$. Since $o_1[s] \cap \text{del}(o_1) = \emptyset$ by definition and both s and $o_1[s]$ are reachable, it follows that for every $f \in (o_1[s] \setminus o_1[\text{pre}(o_1)])$ it holds that $f \cup \text{pre}(o_1) \notin \mathcal{M}^\star$ and $f \cup o_1[\text{pre}(o_1)] \notin \mathcal{M}^\star$. □

Operators with Irreversible Add Effect

The last method is based on the observation that some domains contain facts that once added to a state, they cannot be subsequently deleted by any operator—they are in this sense irreversible. Any fact that does not appear in any delete effect is irreversible and operators that add the same irreversible facts (and nothing else) form pairwise an op-mutex.

Theorem 12. *Let $O \in \mathcal{O}$ be a set of operators with the same add effects, denoted add(O). If for all $o \in \mathcal{O}$, it holds that add$(O) \cap$ del$(o) = \emptyset$ then for every $o_1, o_2 \in O$, $o_1 \neq o_2$, it holds that $\{o_1, o_2\}$ is an op-mutex.*

Proof Sketch. (By contradiction) Let $o_1, o_2 \in O$, $o_1 \neq o_2$, and let $\pi \in \mathcal{P}_\Pi$ s.t. $o_1, o_2 \in \pi$. Assume WLOG, that o_1 occurs before o_2 in π. There is a state s in the plan π where o_2 is applied. Note that since no operator deletes any fact $f \in \text{add}(o_1) = \text{add}(o_2)$ and o_1 has already occurred, then $\text{add}(o_2) \subseteq s$. This means that $o_2[s] \subseteq s$, in contradiction to the strong optimality of the plan. □

The previous op-mutex inference methods produce op-mutexes that are not part of any plan. Theorem 12 can, however, provide op-mutexes that can be a part of some plan (or even an optimal plan), but cannot be part of any strongly optimal plan.

Symmetries

We adopt the definition of symmetry by Shleyfman et al. (2015) with additional stabilizer for the initial state as was proposed by Pochter, Zohar, and Rosenschein (2011).

Definition 13. A **plan preserving symmetry** (or **symmetry** for short) of the transition system $\Theta = \langle \mathcal{S}, L, T, s_I, S_\star \rangle$ is a permutation σ of $\mathcal{S} \cup L$ mapping states to states and labels to labels such that

- $s \xrightarrow{o} s' \in T$ iff $\sigma(s) \xrightarrow{\sigma(o)} \sigma(s') \in T$,
- $\text{c}(o) = \text{c}(\sigma(o))$,
- $s \in S_\star$ iff $\sigma(s) \in S_\star$, and
- $\sigma(s_I) = s_I$

for all states $s, s' \in \mathcal{S}$ and all labels $o \in L$. For notational convenience, we extend permutations σ to sequences $\sigma(\langle x_1, \ldots, x_n \rangle) = \langle \sigma(x_1), \ldots, \sigma(x_n) \rangle$ and sets $\sigma(\{x_1, \ldots, x_n\}) = \{\sigma(x_1), \ldots, \sigma(x_n)\}$.

Symmetries are closed under composition and inverse, and therefore form the automorphism group $\text{Aut}(\Theta)$ of the transition system.

The reason for stabilizing both the initial state and goals (i.e., mapping goals to goals and the initial state to itself) is that under this type of symmetry, symmetries preserve plans in the sense that for any plan, every symmetric sequence of actions is also a plan of the same length and cost.

Theorem 14. *Let Π denote a planning task and let σ be a symmetry for Θ_Π. For every plan π it holds that $\sigma(\pi)$ is also a plan, moreover $|\pi| = |\sigma(\pi)|$ and $\text{c}(\pi) = \text{c}(\sigma(\pi))$.*

Theorem 14 is adopted to our notation and definition of symmetry from the work of Shleyfman et al. (2015, Theorem 1). As a direct corollary of this Theorem we have that $\sigma(\mathcal{P}_\Pi) = \mathcal{P}_\Pi$. This means that op-mutexes are invariant under symmetry.

Proposition 15. *Let Π denote a planning task, and let σ be a symmetry for Θ_Π. If O is an op-mutex so is $\sigma(O)$.*

Proof. (By contradiction) Suppose that $\sigma(O)$ is not an op-mutex, than there exist $\pi \in \mathcal{P}_\Pi$ s.t. $\sigma(O) \in \pi$. By the previous Theorem 14 we have that $\sigma^{-1}(\pi) \in \mathcal{P}_\Pi$. Therefore, $O = \sigma^{-1}(\sigma(O)) \in \sigma^{-1}(\pi)$, contradicting the fact that O is an op-mutex. □

Proposition 15 shows that we can use symmetries to generate more op-mutexes. However, this makes sense only if the input set of op-mutexes was obtained by some method that does not already explore the symmetric op-mutexes. As the work of Röger, Sievers, and Katz (2018) suggests, Proposition 15 would not be useful for h^m heuristics with op-fact compilation, or our methods based on critical-path heuristics (Theorem 11). The same happens for op-mutexes obtained from searching over all (symmetric) operators with irreversible add effects (Theorem 12), for which further inference of op-mutexes using symmetries is not possible.

Abstractions, on the other hand, focus on a subset of the facts of the problem and the results can be extrapolated to other symmetric projections of the problem, e.g., if several mutex groups are symmetric, it may suffice to compute the set of op-mutexes for one of them and then extend the set of op-mutexes with symmetries.

In Proposition 5, we have shown how to find candidates for redundant sets of operators using op-mutex pairs. Symmetries allow us to identify which sets of operators are actually redundant.

Theorem 16. *Let Π denote a planning task, $O_1, O_2 \subseteq \mathcal{O}$ s.t. $O_1 \cap O_2 = \emptyset$, and $\{o_1, o_2\}$ is an op-mutex for every $o_1 \in O_1$ and every $o_2 \in O_2$. If for every $o_2 \in O_2$ there exists $\sigma \in \text{Aut}(\Theta_\Pi)$ s.t. $\sigma(o_2) \in O_1$, then O_2 is redundant.*

Proof. This follows directly from the fact that if there is $o \in O_2$ and $\pi \in \mathcal{P}_\Pi$ s.t. $o \in \pi$, then $\sigma(o) \in \sigma(\pi) \in \mathcal{P}_\Pi$. Hence, since $\sigma(o) \in O_1$, O_2 is redundant by Proposition 5. □

In its simplest form, if two operators form an op-mutex and there exists a symmetry between these two operators, then one of them can be safely removed. Note that Theorem 16 is not restricted to a single symmetry. So, if we find a single operator o and a set of operators that are all symmetric to o and op-mutex with o, then we can safely remove such a set of operators. We can also choose to apply Theorem 16 to a single symmetry. If we find a symmetry σ and a set of operators O such that $\sigma(O) \cap O = \emptyset$ and op-mutexes form a complete bipartite graph between O and $\sigma(O)$, then we can safely remove either O or $\sigma(O)$.

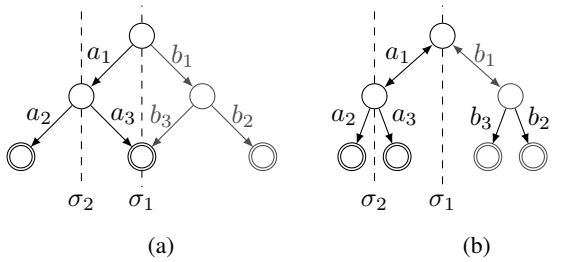

Figure 1: (a) Emergence of a new symmetry; (b) Preserving a symmetry by merging two redundant sets.

Destroying and Preserving Symmetries

In this section, we describe the effect of removing operators on destroying and preserving symmetries. We say that a symmetry is destroyed by removing a certain set of operators if this symmetry is not valid for the reduced planning task. Conversely, a preserved symmetry is the one that still holds in the reduced planning task.

Definition 17. Given an LTS $\Theta = \langle \mathcal{S}, L, T, s_I, S_\star \rangle$ and a set of labels $K \subseteq L$, $\Theta \setminus K = \langle \mathcal{S}, L \setminus K, T_K, s_I, S_\star \rangle$ denotes Θ **reduced by** K, where $s \xrightarrow{l} t \in T_K$ iff $s \xrightarrow{l} t \in T$ and $l \notin K$.

Suppose that there is a label $l_1 \in L \setminus K$ and a transition $s \xrightarrow{l_1} t \in T$, and for some symmetry σ it holds that $\sigma(l_1) \in K$. Then σ is not a symmetry for $\Theta \setminus K$, since then there must exist $s' \xrightarrow{\sigma(l_1)} t' = \sigma(s \xrightarrow{l_1} t)$, but $\Theta \setminus K$ does not contain any such a transition. This is exactly the case for any redundant set obtained according to Theorem 16, because for any such redundant set K there is a symmetry that maps K to $L \setminus K$. Therefore removing K always destroys at least one symmetry.

We can, however, preserve a symmetry if we find a redundant set consisting of operators covering both sides of the symmetry mapping.

Definition 18. Given an LTS Θ and a set of labels $K \subseteq L$, we say that $\sigma \in \mathrm{Aut}(\Theta)$ is **preserved for** K if $\sigma(K) = K$.

Let us show that the term is well defined.

Theorem 19. *Let $\sigma \in \mathrm{Aut}(\Theta)$ be preserved for K, as described above. Then σ is also a symmetry for $\Theta \setminus K$.*

Proof. Note that the transition system $\Theta \setminus K$ differs from Θ only on the sets of transitions T_K and labels $L \setminus K$. Thus, it suffices to show that σ preserves the structure on these sets. Since σ is a permutation of $\mathcal{S} \cup L$, and by definition it holds that $\sigma(\mathcal{S}) = \mathcal{S}$, then $\sigma(L) = L$. Thus, $\sigma(L \setminus K) = \sigma(L) \setminus \sigma(K) = L \setminus K$. This, in turn, implies that σ preserves the transitions in T_K, by definition of the reduced LTS. □

Removal of a redundant set of operators can also result in emergence of new symmetries that were not in the original task. This is illustrated in Fig. 1a: σ_2 is not a symmetry in the LTS, but after removing operators b_i (which are redundant because all $\{a_i, b_i\}$ are symmetric and pairwise op-mutex), σ_2 becomes a symmetry for the resulting reduced LTS.

Fig. 1b shows that a symmetry destroyed by removing redundant operators can be salvaged after subsequently removing another set of redundant operators. If we use symmetry σ_1 to remove b_3 (because $\{a_3, b_3\}$ is an op-mutex), σ_1 is destroyed. If we use afterwards the symmetry σ_2 to remove a_3 ($\{a_2, a_3\}$ is an op-mutex), then σ_1 becomes, again, a symmetry for the reduced planning task.

These observations motivate us to infer redundant operators using a fixpoint computation.

Inference of Redundant Operators

The algorithm for the inference of a redundant set of operators should aim at finding the largest set possible, because, ultimately, we want to use the redundant set for the simplification of the planning task. Redundant sets cannot be automatically merged, but as proven in Proposition 2, a fixpoint computation is possible.

Theorem 16 provides a way to identify redundant sets of operators, but we also already explained that the removal of such operators always destroys at least one symmetry that could be helpful for further inference steps. We propose an algorithm that is based on the assumption that as many symmetries as possible should be preserved in each step in order to increase the chance of finding more redundant operators in the following steps. Moreover, in a fixpoint computation, it can happen that a symmetry that is destroyed in one step can re-emerge again in one of the next steps (Theorem 19), so after removing a set of redundant operators we check which symmetries of the original planning task are preserved under the current set of redundant operators.

Since Theorem 16 requires to find a complete bipartite subgraph of op-mutexes and determining the maximal one is already NP-Hard (Garey and Johnson 1979), we propose a greedy algorithm that gradually increases the size of the resulting redundant set in each step.

The pseudo-code in Algorithm 1 contains two selection steps that can use any rule and the algorithm will still remain

Algorithm 1: Fixpoint computation of a redundant set.

Input: Set of symmetries Σ, set of operators $\mathcal{O}$, set of op-mutex pairs $\mathcal{M}$
Output: Redundant set of operators R

$R \leftarrow \emptyset$;
do
  $\Sigma^+ \leftarrow \{\sigma \in \Sigma \mid \sigma(R) = R\}$;
  $\mathcal{R} \leftarrow \{\texttt{RedundantSet}(\sigma, R) \mid \sigma \in \Sigma^+\}$;
  $R' \leftarrow \texttt{SelectRedundantSet}(\mathcal{R})$;
  $R \leftarrow R \cup R'$;
while $|R'| > 0$;
function `RedundantSet(`σ, R`)`
  $S \leftarrow \emptyset$;
  $C \leftarrow \{o \mid o \in \mathcal{O} \setminus R, o \neq \sigma(o), \{o, \sigma(o)\} \in \mathcal{M}\}$;
  while $|C| > 0$ **do**
    $o' \leftarrow \texttt{SelectCandidate}(C, R \cup S)$;
    $S \leftarrow S \cup \{o'\}$;
    $C \leftarrow \{o \mid o \in C \setminus \{o'\}, \{o, \sigma(o')\} \in \mathcal{M}\}$;
  return S;

domain	#ps	op-mutex pairs in 5 min.				avg. time [s]			op-mutex pairs all finished in 30 min.				
		fam	Π^2_{op}	2-h^2	iadd	fam	Π^2_{op}	2-h^2	#ps	fam	Π^2_{op}	2-h^2	iadd
agricola18	20	(15) 59 862.8	(18) **425 278.3**	(0) 0.0	0.0	**29.1**	86.0	0.0	4	3 478.2	21 065.9	**32 855.5**	0.0
barman11	20	**19.8**	**19.8**	**19.8**	0.0	0.2	**0.0**	0.7	20	**19.8**	**19.8**	**19.8**	0.0
barman14	14	**20.2**	**20.2**	**20.2**	0.0	0.2	**0.0**	1.0	14	**20.2**	**20.2**	**20.2**	0.0
caldera18	20	0.0	235.6	**397.1**	0.0	41.7	**0.3**	31.2	20	0.0	235.6	**397.1**	0.0
cavediving14	20	938.4	**8 331.7**	(16) 1 068.1	140.8	**7.8**	9.1	49.8	16	214.6	847.9	**1 068.1**	18.3
childsnack14	20	**14 562.0**	**14 562.0**	**14 562.0**	213.7	**0.4**	0.8	37.5	20	**14 562.0**	**14 562.0**	**14 562.0**	213.7
floortile11	20	2.0	**3.1**	**3.1**	1.1	0.4	**0.0**	0.9	20	2.0	**3.1**	**3.1**	1.1
floortile14	20	0.9	**1.5**	**1.5**	0.4	0.2	**0.0**	0.2	20	0.9	**1.5**	**1.5**	0.4
hiking14	20	**0.1**	**0.1**	**0.1**	0.0	**0.1**	0.4	13.7	20	**0.1**	**0.1**	**0.1**	0.0
nomystery11	20	20 741.0	20 916.5	(19) **47 163.1**	0.0	9.4	**3.2**	56.8	20	20 741.0	20 916.5	**57 411.9**	0.0
nurikabe18	14	14 747.2	**19 313.1**	(11) 4 584.5	0.0	18.7	**4.6**	16.1	14	14 747.2	19 313.1	**32 476.7**	0.0
openstacks06	30	**3 957.6**	(27) 1 530.5	(23) 457.0	0.0	11.9	**6.7**	24.9	25	448.8	704.7	**958.8**	0.0
openstacks08	30	**340.3**	**340.3**	**340.3**	0.0	0.5	**0.1**	5.5	30	**340.3**	**340.3**	**340.3**	0.0
openstacks11	20	**178.9**	**178.9**	**178.9**	0.0	0.4	**0.1**	3.5	20	**178.9**	**178.9**	**178.9**	0.0
openstacks14	20	**1 078.0**	**1 078.0**	**1 078.0**	0.0	3.0	**0.8**	62.3	20	**1 078.0**	**1 078.0**	**1 078.0**	0.0
organic-synthesis18	18	(10) 0.3	(14) 27.0	(13) **209.6**	0.0	4.0	**0.0**	5.9	11	0.3	3.6	**108.1**	0.0
parcprinter08	30	**3.2**	**3.2**	**3.2**	0.0	0.9	**0.0**	0.1	30	**3.2**	**3.2**	**3.2**	0.0
parcprinter11	20	**2.2**	**2.2**	**2.2**	0.0	0.7	**0.0**	0.1	20	**2.2**	**2.2**	**2.2**	0.0
pathways06	30	**1 184.7**	**1 184.7**	**1 184.7**	1.1	1.0	**0.3**	16.3	30	**1 184.7**	**1 184.7**	**1 184.7**	1.1
pegsol08	30	1.7	1.7	**2.3**	0.0	0.2	**0.0**	0.1	30	1.7	1.7	**2.3**	0.0
pegsol11	20	0.5	0.5	**0.8**	0.0	0.3	**0.0**	0.1	20	0.5	0.5	**0.8**	0.0
petri-net-alignment18	20	(0) 0.0	**276.7**	**276.7**	0.0	0.0	**1.6**	133.1	0	0.0	0.0	0.0	0.0
pipesworld06	50	(33) 21.3	(41) **23.4**	(19) 18.4	0.0	59.8	**35.3**	54.2	26	20.7	**20.8**	**20.8**	0.0
rovers06	40	1 260.4	1 260.4	(27) 90.5	**3 928.4**	**6.4**	10.5	41.2	31	185.9	185.9	185.9	**477.5**
snake18	20	(13) 254.6	**1 096.2**	(0) 0.0	0.0	129.6	**36.1**	0.0	10	198.9	245.0	**255.6**	0.0
sokoban08	30	59.2	60.7	**61.5**	0.0	1.7	**0.1**	7.8	30	59.2	60.7	**61.5**	0.0
sokoban11	20	30.9	31.5	**32.0**	0.0	2.4	**0.1**	5.9	20	30.9	31.5	**32.0**	0.0
spider18	20	(12) 12 660.1	(19) **75 457.6**	(3) 187.8	0.0	**66.3**	73.2	126.5	6	567.4	601.9	**636.0**	0.0
tidybot11	20	1 234.9	**2 205.5**	(9) 12.2	0.0	**5.4**	19.9	30.3	12	0.0	0.0	**35.0**	0.0
tidybot14	20	2 428.7	**4 267.9**	(0) 0.0	0.0	**10.7**	38.2	0.0	8	82.7	144.3	**244.9**	0.0
tpp06	30	(27) 1 901.1	**3 782.2**	(20) 180.1	0.0	31.0	**4.3**	21.8	25	1 056.1	1 056.1	**1 117.4**	0.0
trucks06	30	1 684.4	1 693.3	**2 127.2**	1 498.9	0.6	**0.4**	28.6	30	1 684.4	1 693.3	**2 127.2**	1 498.9
woodworking08	30	48.9	49.7	**49.8**	0.0	0.5	**0.0**	1.2	30	48.9	49.7	**49.8**	0.0
woodworking11	20	29.3	30.0	**30.1**	0.0	0.5	**0.0**	1.0	20	29.3	30.0	**30.1**	0.0
overall	806	(738) 139 255.6	(787) **583 264.2**	(644) 74 342.9	5 784.5	11.5	**9.4**	21.0	672	60 988.9	84 602.7	147 469.5	2 211.0

Table 1: Left: Number of inferred op-mutex pairs (in thousands) with a time limit of 5 minutes. If the method did not terminate in time, the number of successfully processed tasks is shown in parentheses. Middle: Average time per instance in seconds. Right: Number of inferred op-mutex pairs (in thousands) in commonly solved problems with a time limit of 30 minutes.

sound. `SelectRedundantSet` on line 5 selects one of the redundant sets found on the previous line. In our implementation, we select the largest of those sets that preserves the most symmetries. `SelectCandidate` on line 12 selects one operator from the set of candidates C. We select the operator $o' \in C$ for which the set $R \cup S \cup \{o'\}$ preserves the most symmetries.

Proposition 20. *Algorithm 1 always returns a redundant set of operators.*

Proof Sketch. The main cycle (lines 2–7) computes a union of sets found by `RedundantSet` function in consecutively reduced planning tasks. Therefore what remains to show is that these sets are redundant (Proposition 2). To prove that `RedundantSet` always returns a redundant set, it is enough to show that in every cycle (a) $\{o, \sigma(o)\}$ is op-mutex for every $o \in C$; (b) $\{o, o'\}$ is op-mutex for every $o \in S$ and every $o' \in C$; (c) $S \cup \{o'\}$ is redundant for any $o' \in C$. (a) follows from line 10. (b) follows from line 14 and the symmetry of op-mutexes (Proposition 15). (c) follows from (a), (b), and Theorem 16 with sets $O_1 := \sigma(S)$ and $O_2 := S$ and a single symmetry σ. □

Experimental Results

The proposed methods were implemented in C and the source code is publicly available[1]. Symmetries are computed using the BLISS library (Junttila and Kaski 2007) on the Problem Description Graph (PDG) of the task (Pochter, Zohar, and Rosenschein 2011; Shleyfman et al. 2015). We used all IPC benchmarks 2006–2018 from the optimal track. Since our methods are described for STRIPS planning tasks without conditional effects, we compile them away (Nebel 2000). We exclude all instances where the compilation of conditional effects or grounding of the task exceeded the time or memory limits. The experiments were performed on a cluster with Intel E5-2670 2.6 GHz processor with 8 GB memory limit for each task.

We evaluated four methods for inference of op-mutexes: `fam` refers to the method based on abstractions (Theorem 6) used on projections to individual maximal fact-alternating mutex groups (fam-groups) (Fišer and Komenda 2018); Π^2_{op} refers to h^2 reachability analysis of the op-fact compilation (Theorem 8); `2-h`^{2} refers to running reachability on the 2-$\mathcal{M}$ compilation (Definition 10) with mutexes $\mathcal{M}$ obtained from the h^2 reachability with all operators (Theorem 11); and `iadd` refers to looking for operators with a single irreversible add effect (Theorem 12).

Table 1 summarizes the results on the number of inferred op-mutex pairs (in thousands). We list only the domains in which at least one op-mutex pair was found. The results show that it is possible to infer a significant number of op-mutex pairs in many different domains (34 out of 83) even with a time limit of 5 minutes which is suitable for a pre-

[1] https://gitlab.com/danfis/cplan.git, branch aaai19

domain	#ps	h^2+de	fam	Π^2_{op}	2-h^2	h^2+de combined with fam	h^2+de combined with Π^2_{op}	h^2+de combined with 2-h^2
barman11	20	**52.11**	4.71	24.48	20.65	**+3.60**	**+3.60**	**+3.60**
barman14	14	**53.43**	4.94	25.28	21.25	**+4.01**	**+4.01**	**+4.01**
caldera18	19	**39.34**	0.73	13.25	12.64	0.00	**+1.45**	+0.72
cavediving14	5	0.66	**1.15**	**1.44**	0.55	**+1.15**	**+1.15**	+0.55
childsnack14	20	0.00	**39.58**	**39.58**	**39.58**	**+39.58**	**+39.58**	**+39.62**
hiking14	11	0.00	**0.07**	**0.07**	**0.07**	**+0.07**	**+0.07**	**+0.07**
parcprinter08	24	**70.68**	28.61	28.61	28.61	0.00	0.00	0.00
parcprinter11	17	**70.15**	25.75	25.75	25.75	0.00	0.00	0.00
pathways06	30	**3.94**	2.22	2.26	2.26	**+3.73**	**+3.73**	**+3.73**
pegsol08	9	**13.57**	0.00	1.92	1.92	**+0.36**	**+0.36**	**+0.36**
pegsol11	8	**9.53**	1.08	3.51	3.45	**+0.20**	**+0.20**	**+0.20**
pipesworld06	17	**11.46**	0.04	4.04	4.04	**+0.04**	**+0.04**	**+0.04**
scanalyzer08	11	**3.19**	0.00	2.10	2.10	0.00	0.00	0.00
scanalyzer11	8	**3.17**	0.00	2.12	2.12	0.00	0.00	0.00
sokoban08	6	0.24	1.54	**1.71**	1.65	+1.54	**+1.60**	**+1.60**
sokoban11	4	0.35	1.86	**2.12**	2.04	+1.86	**+1.95**	**+1.95**
tpp06	15	**37.96**	1.49	19.97	18.97	**+1.79**	**+1.79**	+1.49
trucks06	12	**75.02**	0.00	0.71	0.71	**+7.87**	**+7.87**	**+7.87**
woodworking08	24	**53.47**	2.83	10.79	10.79	**+1.60**	**+1.60**	**+1.60**
woodworking11	18	**53.72**	2.73	10.17	10.17	**+1.83**	**+1.83**	**+1.83**
overall	292	**53.72**	7.99	13.73	13.17	+4.29	**+4.39**	+4.32

Table 2: Average percentage of removed operators in each domain. The column #ps shows the number of instances in which all methods successfully terminated before the 15 minutes time limit and at least one operator was pruned by at least one method; the left part shows the average percentage of removed operators in those instances; and the right part shows the increase when h^2+de is combined with the methods using op-mutexes. The last row (overall) shows averages over all instances. Maximums are highlighted in bold.

domain	#ps	base	fam	Π^2_{op}	2-h^2
agricola18	17	251306	(15) **−5 108**	(12) −3 242	(0) 0
barman11	20	7408	**−554**	**−554**	**−554**
barman14	14	6010	**−522**	**−522**	**−522**
caldera18	20	24178	0	**−492**	−220
cavediving14	20	91832	**−74**	**−74**	(16) −37
childsnack14	20	53698	**−21 660**	**−21 660**	−21 624
hiking14	20	35822	**−11**	**−11**	**−11**
organic-synthesis18	13	47614	(12) −27	−432	(12) **−1 095**
pathways06	30	40595	**−731**	**−731**	**−731**
pegsol08	30	4392	**−6**	**−6**	**−6**
pegsol11	20	3320	**−3**	**−3**	**−3**
pipesworld06	42	1122448	(37) −69	**−121**	(22) −27
sokoban08	30	12637	−73	**−74**	**−74**
sokoban11	20	7139	−45	**−46**	**−46**
tpp06	30	107409	(25) −1 147	**−2 167**	(22) −429
trucks06	21	22769	−4 673	−4 673	**−4 679**
woodworking08	30	12535	**−400**	**−400**	**−400**
woodworking11	20	8159	**−307**	**−307**	**−307**
Σ	417	1859271	−35 410 (404)	**−35 515** (412)	−30 765 (367)

Table 3: Number of operators pruned over the baseline h^2+de within a time limit of 15 minutes. The column #ps shows the number of instances in which at least one method successfully finished within the time limit. For methods that finished in less instances, their number of successfully finished instances is indicated in parenthesis.

processing step. The average time of iadd is omitted because in all cases it was a fraction of a second. The most successful method was Π^2_{op} mainly because it offers a good trade-off between op-mutex pairs found and computational effort. The reason why fam was slower than Π^2_{op} on average is that in some cases the inference of fam-groups is slow. As expected, the slowest of all methods was 2-h^2, because it requires to compute h^2 reachability for every operator.

We also directly compare the op-mutexes found by all methods on commonly solved instances under 30 minutes (see right part of Table 1). In every case, the set of op-mutexes found by fam was a subset of those found by Π^2_{op}, and Π^2_{op} found a subset of 2-h^2. Therefore there is a promising potential in 2-h^2, if the computation can be sped-up (e.g. by using symmetries in the computation of h^2 (Röger, Sievers, and Katz 2018)).

The implementation of Algorithm 1 was experimentally evaluated as a standalone preprocessing step and in combination with the pruning using forward/backward h^2 (Alcázar and Torralba 2015) and the detection of dead-end operators (Fišer and Komenda 2018) (we will refer to the combination of the last two as h^2+de). Table 2 shows the average percentage of removed operators within each domain and over all evaluated instances. Only the instances in which all methods successfully finished within the 15 minutes time limit are listed. In all domains except cavediving, childsnack, hiking, and sokoban, h^2+de prunes more operators than any of our methods, but combining h^2+de with our methods further increases the number of removed operators as shown in the right part of the table.

Table 3 encapsulates the results from combining h^2+de with our pruning methods in absolute numbers. We set the baseline (the column base) as the number of operators after the grounding and h^2+de pruning. The remaining columns contain the number of additionally removed operators after a subsequent application of Algorithm 1 using the op-mutexes found by our methods and h^2+de pruning until a fixpoint is reached. We do not combine fam, Π^2_{op}, and 2-h^2 methods because of the observed dominance between them. We also exclude iadd, because by itself it managed to prune only a third of the operators of that of the remaining three methods, and if combined with any of the remaining three, the results were, surprisingly, worse, probably because of the greedy selection steps in Algorithm 1. The results show that op-mutexes can be combined with symmetries to simplify planning tasks in several domains. In some domains, the reduction in the number of operators is quite significant, especially in childsnack, agricola, trucks, and barman. Finding op-mutexes with Π^2_{op} achieves the best results in most cases.

We also tried the most promising Π^2_{op} pruning technique with the Fast Downward planner (Helmert 2006). We used A* with the LM-Cut (lmc) heuristic (Helmert and Domshlak 2009), the merge-and-shrink (m&s) heuristic with SCC-DFP merge strategy and non-greedy bisimulation shrink strategy (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2016), and the potential (pot) heuristic optimized for all syntactic states (Seipp, Pommerening, and Helmert 2015). The time limit was set to 30 minutes for the whole planning process.

When we set the Π^2_{op} pruning phase to abstain from the inference of op-mutexes if no symmetries were found, the planners solved all problems that were solved without Π^2_{op} except of one problem in organic-synthesis for all three heuristics, and one problem in scanalyzer for lmc. In these cases, the inference of op-mutexes took too long (organic-synthesis) or exceeded the memory (scanalyzer) limit. However, despite the size of many instances is reduced by the pruning, this was not substantially reflected on the cover-

age. Only in the agricola domain, pot with Π^2_{op} was able to solve two more problems, and m&s one more problem. The reductions had also a negligible effect on the number of expanded states in most instances. In childsnack, which is the domain where more operators are pruned by our method, we measured about twice as many expanded states per second by lmc with Π^2_{op}. However, no planner solved any instance in this domain.

Conclusion

We introduced a new notion of strong operator mutexes (op-mutexes) as sets of operators that cannot be part of the same strongly optimal plan and proposed four different methods for inference of op-mutexes. We proved that every op-mutex contains at least one operator that can be safely removed from the planning task and we have experimentally evaluated that they can be found in a sizable amount of planning domains. Combining op-mutexes with structural symmetries provides further information about which operators can actually be removed, and we showed that there are some domains where removing operators yields a significant reduction in the size of the planning tasks. Even though our experiments show that this reduction does not translate into significant gains for heuristic search planners, this opens new avenues of research on how to leverage operator mutexes to simplify planning tasks.

Acknowledgements

The work of Daniel Fišer was supported by the Czech Science Foundation (grant no. 18-24965Y and 18-07252S). The work of Alexander Shleyfman was supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures".

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proc. ICAPS'15*, 2–6.

Culberson, J. C., and Schaeffer, J. 1996. Searching with pattern databases. In *Canadian Conference on AI*, volume 1081 of *Lecture Notes in Computer Science*, 402–416. Springer.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proc. ICAPS'12*, 343–347.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In *Proc. ICAPS'13*, 298–302.

Edelkamp, S. 2001. Planning with pattern databases. In *Proc. ECP'01*, 13–24.

Fišer, D., and Komenda, A. 2018. Fact-alternating mutex groups for classical planning. *Journal of Artificial Intelligence Research* 61:475–521.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *Proc. IJCAI'99*, 956–961.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman.

Gerevini, A., and Schubert, L. 1998. Inferring state-constraints for domain independent planning. In *Proc. AAAI'98*, 905–912.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proc. AIPS'00*, 140–149.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In *Proc. ICAPS'09*, 354–357.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. ICAPS'09*, 162–169.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery* 61(3):16.1–16.63.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proc. ICAPS'07*, 176–183.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc. ALENEX'07*, 135–149. SIAM.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research* 12:271–315.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In *Proc. AAAI'11*.

Riddle, P. J.; Barley, M. W.; Franco, S.; and Douglas, J. 2015. Automated transformation of PDDL representations. In *Proc. SOCS'15*, 214–215.

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. In *Proc. AAAI'00*, 806–811.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In *Proc. ICAPS'03*, 32–41.

Röger, G.; Sievers, S.; and Katz, M. 2018. Symmetry-based task reduction for relaxed reachability analysis. In *Proc. ICAPS'18*, 208–217.

Seipp, J., and Helmert, M. 2018. Counterexample-guided Cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research* 62:535–577.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New optimization functions for potential heuristics. In *Proc. ICAPS'15*, 193–201.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and symmetries in classical planning. In *Proc. AAAI'15*, 3371–3377.

Sievers, S.; Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Factored symmetries for merge-and-shrink abstractions. In *Proc. AAAI'15*, 3378–3385.

Sievers, S.; Wehrle, M.; Helmert, M.; and Katz, M. 2017. Strengthening canonical pattern databases with structural symmetries. In *Proc. SOCS'17*, 91–99.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In *Proc. AAAI'14*, 2358–2366.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An analysis of merge strategies for merge-and-shrink heuristics. In *Proc. ICAPS'16*, 294–298.