Symmetry Breaking in Star-Topology Decoupled Search

Daniel Gnad and **Álvaro Torralba**
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
{gnad,torralba}@cs.uni-saarland.de

Alexander Shleyfman
Technion - Israel Institute of Technology
Industrial Engineering & Management
Haifa, Israel
alesh@campus.technion.ac.il

Jörg Hoffmann
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
hoffmann@cs.uni-saarland.de

Abstract

Symmetry breaking is a well-known method for search reduction. It identifies state-space symmetries prior to search, and prunes symmetric states during search. A recent proposal, star-topology decoupled search, is to search not in the state space, but in a factored version thereof, which avoids the multiplication of states across leaf components in an underlying star-topology structure. We show that, despite the much more complex structure of search states – so-called *decoupled states* – symmetry breaking can be brought to bear in this framework as well. Starting from the notion of structural symmetries over states, we identify a sub-class of such symmetries suitable for star-topology decoupled search, and we show how symmetries from that sub-class induce symmetry relations over decoupled states. We accordingly extend the routines required for search pruning and solution reconstruction. The resulting combined method can be exponentially better than both its components in theory, and this synergetic advantage is also manifested in practice: empirically, our method reliably inherits the best of its base components, and often outperforms them both.

Introduction

We venture to integrate two methods for search reduction in forward state space search, *symmetry breaking* and *star-topology decoupled search* (DS). The former is well-established and well-explored across several CS sub-areas, including classical planning (e. g. (Starke 1991; Emerson and Sistla 1996; Fox and Long 1999; Rintanen 2003; Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012)). It allows to prune (parts of) the exponential search resulting from, e. g., the presence of objects with symmetric behavior. Star-topology decoupled search, on the other hand, has only recently been invented, in classical planning (Gnad and Hoffmann 2015; Gnad, Hoffmann, and Domshlak 2015). It avoids the enumeration of states across leaf factors in a star topology, and has been shown to dramatically improve performance in pronounced star topologies with many leaf factors. Yet symmetric behaviors may cause exponential search in this setting as well, so the question is whether the two methods can be integrated. We answer this question in the affirmative, and we demonstrate the theoretical and practical benefits.

We focus on classical planning. Symmetry breaking in this setting has been first proposed by Fox and Long (1999) in the context of Graphplan (Blum and Furst 1997; Long and Fox 1999), the basic idea being to recognize symmetric objects – ones that behave symmetrically – in the PDDL input, forming groups which the search keeps track of, and which the basic search steps exploit by considering only one object from each group. Variants of symmetry breaking have later been proposed for SAT-based planning (Rintanen 2003) and, most recently, for forward state space search (Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012; 2013; Wehrle et al. 2015; Sievers et al. 2015; Shleyfman et al. 2015). In the latter setting, the symmetries take the form of symmetry groups across states. If several states from a group are encountered, only one of these is explored; upon finding a goal state, a reconstruction procedure takes care of any discontinued paths in the solution.

Star-topology decoupled search, short *decoupled search*, is a form of factored planning (e. g. (Knoblock 1994; Amir and Engelhardt 2003; Brafman and Domshlak 2006; Kelareva et al. 2007; Brafman and Domshlak 2008; 2013; Fabre et al. 2010)). The *factors* – disjoint subsets of state variables – are required to form a star, where a single *center* factor has arbitrary dependencies with each of a set of *leaf* factors, yet no two leaf factors depend upon each other directly. As prior work showed (Gnad and Hoffmann 2015; Gnad, Hoffmann, and Domshlak 2015), this kind of structure can be exploited by searching over center paths only (transitions paths affecting the center), maintaining the possible moves given such a path separately for each leaf. In catchy though imprecise analogy to graphical models, decoupled search "instantiates the center to break the conditional dependencies between leaves".

Symmetry breaking and decoupled search are clearly orthogonal, in the sense that the structures exploited by either – symmetrical behavior vs. conditional independence of leaf factors – are, in general, different. However, we cannot simply "switch both methods on", as the structure of search states in decoupled search, so-called *decoupled states*, is much more complex, containing exhaustive tracking information for each leaf factor. *How can symmetry breaking be brought to bear in this new search structure?*

This question is relevant as decoupled search may still suffer from symmetrical behavior, within the center and across leaves. To answer the question, we start from Shleyfman et al.'s (2015) *structural symmetries*, which capture previously proposed concepts of symmetry in classical planning, and which can be derived from an input task's syntax in a simple declarative manner. We identify a sub-class of structural symmetries suitable for decoupled search (essentially, not interfering with the star topology). We show how to find such symmetries, how they induce symmetry relations over decoupled states, and how the routines for search pruning and solution reconstruction can be extended. The resulting combined method can be exponentially better than each of its components. Indeed, there are cases where it is exponentially better than both, showing synergistic effects where the combination is "more than the sum of its components". Such synergy is also manifested in practice: empirically, our method reliably inherits the best of its components, outperforms each component overall, and in several individual domains outperforms them both. For space reasons, some proofs are deferred to a TR (Gnad et al. 2017).

Preliminaries

We use finite-domain state variables (FDR) (Bäckström and Nebel 1995; Helmert 2006). A *planning task* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, where $\mathcal{V}$ is a set of *variables*, each associated with a finite domain $\mathcal{D}(v)$. We identify (partial) variable assignments with sets of variable/value pairs, written as $\langle var, val \rangle$. A *state* is a complete assignment to $\mathcal{V}$. By S we denote the set of all states of Π. I is the *initial state* of Π. The *goal* G is a partial assignment to $\mathcal{V}$. For a partial assignment p, we denote with $vars(p) \subseteq \mathcal{V}$ the subset of variables on which p is defined. For $V \subseteq vars(p)$, by $p[V]$ we denote the assignment to V made by p. We say that a (partial) assignment p *satisfies* a condition q, denoted $p \models q$, if $vars(q) \subseteq vars(p)$, and $p[v] = q[v]$ for all $v \in vars(q)$. $\mathcal{A}$ is a finite set of *actions*, each a triple $\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle$ of *precondition*, *effect*, and *cost*, where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are partial assignments to $\mathcal{V}$, and $\mathsf{cost}(a) \in \mathbb{R}^{0+}$. An action a is *applicable* in a state s if $s \models \mathsf{pre}(a)$. Applying a in s changes the value of all $v \in vars(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$, and leaves s unchanged elsewhere. The outcome is denoted $s[\![a]\!]$.

The *state space* of Π is denoted $\mathcal{T}_\Pi$. We will sometimes write $s \xrightarrow{a} t$ for a transition from s to t with action a. A *plan* for Π is an action sequence π iteratively applicable in I and ending in s_G s.t. $s_G \models G$. The plan is *optimal* if its summed-up cost, denoted $\mathsf{cost}(\pi)$, is minimal. Given a state s, we denote the cost of an optimal plan for s as $h^*(s)$.

Given a directed graph $\mathcal{G} = \langle N, E \rangle$, a permutation σ on the vertices N, s.t. $(n, n') \in E$ iff $(\sigma(n), \sigma(n')) \in E$, is called an *automorphism*. The automorphisms of a graph $\mathcal{G}$ form a group under composition. We call this group an *automorphism group*, and denote it by $Aut(\mathcal{G})$. For every permutation $\sigma \in Aut(\mathcal{G})$ there exists an inverse permutation $\sigma^{-1} \in Aut(\mathcal{G})$ s.t. $\sigma \circ \sigma^{-1}$ is the identity permutation.

Base Methods

We start from two base methods, symmetry breaking and decoupled search. In what follows, we summarize these methods to the extent required to describe our modifications.

Symmetry Breaking

Symmetry breaking considers equivalence classes of symmetrical states in the search space, and allows for using representative states of each equivalence class. Recently, Shleyfman et al. (2015) introduced the notion of *structural symmetries*, which capture previously proposed concepts of symmetry breaking for classical planning. In a nutshell, structural symmetries are a relabeling of the factored representation of a given planning task Π. Actions are mapped to actions, variables to variables, and values to values (preserving the variable/value pairs structure). This relabeling induces an automorphism of the state space $\mathcal{T}_\Pi$. Herein, we follow the definition of structural symmetries for FDR planning tasks as defined by Wehrle et al. (2015).

Definition 1 (Structural Symmetry) *For a planning task* $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$*, let* P *be the set of* Π*'s* facts*, i. e., pairs* $\langle v, d \rangle$ *with* $v \in \mathcal{V}$ *and* $d \in \mathcal{D}(v)$*. A* structural symmetry *for* Π *is a permutation* $\sigma : P \cup A \to P \cup \mathcal{A}$ *such that:*

1. $\sigma(P) = P$*, where* $P := \{\{\langle v, d \rangle \mid d \in \mathcal{D}(v)\} \mid v \in \mathcal{V}\}$.
2. $\sigma(\mathcal{A}) = \mathcal{A}$*, and, for all* $a \in \mathcal{A}$*,* $\sigma(\mathsf{pre}(a)) = \mathsf{pre}(\sigma(a))$*,* $\sigma(\mathsf{eff}(a)) = \mathsf{eff}(\sigma(a))$*, and* $\mathsf{cost}(\sigma(a)) = \mathsf{cost}(a)$.
3. $\sigma(G) = G$.

Here, for a set X *we define* $\sigma(X) := \{\sigma(x) \mid x \in X\}$ *(note that this notation can be applied recursively). For a partial state* s*,* $s' := \sigma(s)$ *is the partial state obtained from* s *such that for all* $\langle v, d \rangle$ *with* $v \in vars(s)$ *and* $d \in \mathcal{D}(v)$*,* $\sigma(\langle v, d \rangle) = \langle v', d' \rangle$ *and* $s'[v'] = d'$.

A set of structural symmetries Σ for a planning task Π induces a subgroup Γ of the automorphism group $Aut(\Pi)$, which in turn defines an equivalence relation over the states S of Π. Namely, we say that s is *symmetric* to s' iff there exists an automorphism $\sigma \in \Gamma$ such that $\sigma(s) = s'$.

Forward search algorithms with symmetry elimination do not consider all states $s \in S$, but only a single representative element of the equivalence class of s. These equivalence classes are called *orbits* and are usually represented by one of its member states that is called the *canonical* state. A* with symmetry elimination then explores all applicable actions, and prunes the resulting successor states if another representative of their orbit has already been encountered during search. Due to the properties of structural symmetries, this reduced state transition graph is guaranteed to still contain an optimal plan in s. However, determining if two states are symmetric is **NP**-hard (Luks 1993). To overcome this, one can perform symmetry elimination by computing an approximated canonical representative with an incomplete ad-hoc procedure that is not guaranteed to detect all symmetries (Pochter, Zohar, and Rosenschein 2011). We use the *orbit space search* (OSS) algorithm introduced by Domshlak, Katz, and Shleyfman (2015). OSS replaces each state by its approximated canonical representative, resulting in a search over the state transition graph induced by the (approximated) canonical states.

Star-Topology Decoupled Search

Decoupled search (DS) is a technique developed to avoid the combinatorial explosion of having to enumerate all possible variable assignments of causally independent parts of a planning task. It does so by partitioning the state variables into a *factoring* $\mathcal{F}$, whose elements are called *factors*. By imposing a structural requirement on the interaction between these factors, namely a *star topology*, decoupled search can efficiently handle cross-factor dependencies. A *star factoring* is one that has a *center* $F^C \in \mathcal{F}$ that interacts arbitrarily with all other factors $F^L \in \mathcal{F}^L := \mathcal{F} \setminus F^C$, called *leaves*, but where the only interaction between leaves is via the center.

Actions affecting F^C are called *center actions*, denoted $\mathcal{A}^C$, and those affecting a leaf F^L are called *leaf actions*, denoted $\mathcal{A}^L$. A sequence of center actions applicable to I in the projection onto F^C is a *center path*, a sequence of leaf actions affecting F^L, applicable to I in the projection onto F^L, is a *leaf path*. A complete assignment to F^C is called a *center state*, an assignment to an $F^L \in \mathcal{F}^L$ is called a *leaf state*. The set of all leaf states is denoted S^L, and that of a particular leaf F^L is denoted $S^L|_{F^L}$. We define the set of leaf actions *enabled* in a center state s^C as $\mathcal{A}^L|_{s^C} := \{a^L \mid a^L \in \mathcal{A}^L \wedge s^C \models \mathsf{pre}(a^L)[F^C]\}$.

A *decoupled state* $s^\mathcal{F}$ is a pair $\langle \mathsf{center}(s^\mathcal{F}), \mathsf{prices}(s^\mathcal{F})\rangle$, where $\mathsf{center}(s^\mathcal{F})$ is a center state, and $\mathsf{prices}(s^\mathcal{F}) : S^L \mapsto \mathbb{R}^{0+} \cup \{\infty\}$ is the *pricing function*, that assigns every leaf state a non-negative price. The pricing function is maintained during decoupled search in a way so that the price of a leaf state s^L is the cost of a cheapest leaf path that ends in s^L and that is *compliant*, i. e., that can be scheduled alongside the center path executed up to $s^\mathcal{F}$.

By $S^\mathcal{F}$ we denote the set of all decoupled states. We say that a decoupled state $s^\mathcal{F}$ *satisfies* a condition p, denoted $s^\mathcal{F} \models p$, iff (i) $\mathsf{center}(s^\mathcal{F}) \models p[F^C]$ and (ii) for every leaf $F^L \in \mathcal{F}^L$ there exists $s^L \in S^L|_{F^L}$ s.t. $s^L \models p[F^L]$ and $\mathsf{prices}(s^\mathcal{F})[s^L] < \infty$. The *initial decoupled state* $I^\mathcal{F}$ is defined as $I^\mathcal{F} := \langle \mathsf{center}(I^\mathcal{F}), \mathsf{prices}(I^\mathcal{F})\rangle$, where $\mathsf{center}(I^\mathcal{F}) = I[F^C]$. The pricing function is given, for each $F^L \in \mathcal{F}^L$, as $\mathsf{prices}(I^\mathcal{F})[s^L_0] = 0$ where $s^L_0 = I[F^L]$; and elsewhere as $\mathsf{prices}(I^\mathcal{F})[s^L] = c(s^L_0)$ where $c(s^L_0)$ is the cost of a cheapest path of $\mathcal{A}^L|_{\mathsf{center}(I^\mathcal{F})} \setminus \mathcal{A}^C$ actions from s^L_0 to s^L. If no such path exists, then $c(s^L_0) = \infty$. The set of *decoupled goal states* $S^\mathcal{F}_G$ is $S^\mathcal{F}_G := \{s^\mathcal{F}_G \mid s^\mathcal{F}_G \models G\}$.

In decoupled search, only center actions are applied. A center action a^C is *applicable* in a decoupled state $s^\mathcal{F}$ if $s^\mathcal{F} \models \mathsf{pre}(a^C)$. By S^L_{compl} we define the set of leaf states that satisfy the leaf precondition of a^C, i. e., $S^L_{compl} := \{s^L \mid s^L \models \mathsf{pre}(a^C)[F^L] \wedge \mathsf{prices}(s^\mathcal{F})[s^L] < \infty\}$. Applying a^C to $s^\mathcal{F}$ results in the decoupled state $t^\mathcal{F} = s^\mathcal{F}[\![a^C]\!]$ as follows: $\mathsf{center}(t^\mathcal{F}) := \mathsf{center}(s^\mathcal{F})[\![a^C]\!]$, $\mathsf{prices}(t^\mathcal{F})[t^L] := \min_{s^L \in S^L_{compl}}(\mathsf{prices}(s^\mathcal{F})[s^L] + c(u^L))$ where $s^L[\![a^C]\!] = u^L$, and $c(u^L)$ is the cost of a cheapest path of $\mathcal{A}^L|_{\mathsf{center}(t^\mathcal{F})} \setminus \mathcal{A}^C$ actions from u^L to t^L if such a path exists, else $c(u^L) = \infty$. A *decoupled plan* for Π is a sequence of center actions $\pi^\mathcal{F}$ from $I^\mathcal{F}$ to some $s^\mathcal{F}_G \in S^\mathcal{F}_G$. The *global plan* corresponding to $\pi^\mathcal{F}$, $GlobalPlan(\pi^\mathcal{F})$, is constructed by augmenting $\pi^\mathcal{F}$ with cheapest-compliant leaf paths, i. e., leaf action sequences that lead to the pricing function of $s^\mathcal{F}_G$.

A decoupled state $s^\mathcal{F}$ can be interpreted as a set of explicit states. This set takes the form of a *hypercube* whose dimensions are the leaf factors F^L. Formally, such a hypercube is defined as follows:

Definition 2 (Hypercube) *Let Π be a planning task, and $\mathcal{F}$ a star factoring. Then a state p in Π is a* member state *of a decoupled state $s^\mathcal{F}$, if $p[F^C] = \mathsf{center}(s^\mathcal{F})$ and, for all leaves $F^L \in \mathcal{F}^L$, $\mathsf{prices}(s^\mathcal{F})[p[F^L]] < \infty$. We say that p has* cost *$cost_{s^\mathcal{F}}(s)$ in $s^\mathcal{F}$, where $cost_{s^\mathcal{F}}(s) := \sum_{F^L \in \mathcal{F}^L} \mathsf{prices}(s^\mathcal{F})[p[F^L]]$. The* hypercube *of $s^\mathcal{F}$, denoted $[s^\mathcal{F}]$, is the set of all member states of $s^\mathcal{F}$.*

The hypercube of $s^\mathcal{F}$ captures both, the reachability and the prices of all member states p of $s^\mathcal{F}$. We define the goal distance of a decoupled state $s^\mathcal{F}$ as $h^*(s^\mathcal{F}) := min_{s \in [s^\mathcal{F}]} cost_{s^\mathcal{F}}(s) + h^*(s)$.

Symmetry Relations over Decoupled States

To define symmetries in the decoupled state space, we restrict the allowed class of structural symmetries by imposing additional requirements on their properties.

Definition 3 (Decoupled Structural Symmetry) *Let Π be an FDR task, and let $\mathcal{F}$ be a star factoring. Then σ is a* decoupled structural symmetry *if and only if σ is a structural symmetry and in addition it holds that:*

(i) $\sigma(F^C) = F^C$.
(ii) $\forall F^L \in \mathcal{F}^L : \sigma(F^L) \in \mathcal{F}^L$.

In other words, decoupled structural symmetries are the subset of structural symmetries that (i) stabilize the center (center facts are only mapped to center facts), and (ii) stabilize the leaves (when permuting a fact of a leaf F^{L_1} to a fact of F^{L_2}, all facts of F^{L_1} must be permuted to facts of F^{L_2}). Note that property (i) follows from property (ii) and the fact that σ is a structural symmetry. Nevertheless, we include property (i) into the definition for better readability. These properties are not tautological, i. e., there exist structural symmetries that do not satisfy them:

Proposition 1 *Not every structural symmetry is a decoupled structural symmetry.*

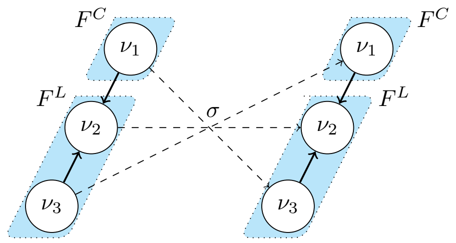

Figure 1: A structural symmetry σ that is not a decoupled symmetry for the depicted factoring.

An example is shown in Figure 1. Such symmetries cannot be exploited in decoupled search. We remark though that, in practice as far as reflected by the IPC benchmarks and our current factoring strategies, this is not a serious limitation: all structural symmetries found are in fact decoupled structural symmetries.

Applying a decoupled permutation to a decoupled state requires permuting its center state and pricing function. We

$$\begin{tikzcd} s \in [s^{\mathcal{F}}] \arrow[r, dashed, "\sigma"'] \arrow[d, leftrightarrow] & \sigma(s) \in [\sigma(s^{\mathcal{F}})] \arrow[d, leftrightarrow] \\ s^{\mathcal{F}} \arrow[r, dashed, "\sigma"'] & \sigma(s^{\mathcal{F}}) \end{tikzcd}$$

Figure 2: How applying a permutation to a decoupled state corresponds to applying it on its hypercube.

obtain the permuted pricing function by assigning the price of each leaf state s^L to $\sigma(s^L)$:

Definition 4 (Permuted Decoupled States) *Let $s^{\mathcal{F}}$ be a decoupled state, and let σ be a decoupled structural symmetry. We define $\sigma(s^{\mathcal{F}})$ as a decoupled state $\langle \sigma(\mathsf{center}(s^{\mathcal{F}})), \sigma(\mathsf{prices}(s^{\mathcal{F}}))\rangle$ where $\sigma(\mathsf{prices}(s^{\mathcal{F}})) = \mathsf{prices}(\sigma(s^{\mathcal{F}}))$, and for each leaf state $s^L \in S^L$ $\mathsf{prices}(\sigma(s^{\mathcal{F}}))[s^L] = \mathsf{prices}(s^{\mathcal{F}})[\sigma^{-1}(s^L)]$.*

Note that $\sigma(s^{\mathcal{F}})$ is a valid decoupled state thanks to the properties of decoupled structural symmetries: Property (i) ensures that $\sigma(\mathsf{center}(s^{\mathcal{F}}))$ is indeed a center state, because center variables are only mapped to center variables. Property (ii) ensures that leaf states are always mapped to leaf states, so $\mathsf{prices}(s^{\mathcal{F}})[\sigma^{-1}(s^L)]$ is always well defined. Additionally, (ii) ensures that all states in a leaf F^{L_1} are permuted into states of the same target leaf F^{L_2}, so we cannot end up with a leaf factor that does not have any finite price. In fact, it can be shown that the prices of $\sigma(s^{\mathcal{F}})$ correspond to those obtained on a path from $\sigma(I^{\mathcal{F}})$ to $\sigma(s^{\mathcal{F}})$.

Proposition 2 *Let Π be an FDR task, and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}$ be a decoupled state reachable from $I^{\mathcal{F}}$, and let σ be a decoupled structural symmetry. Then $\sigma(s^{\mathcal{F}})$ is a decoupled state reachable from $\sigma(I^{\mathcal{F}})$ s.t. for each leaf state $s^L \in S^L$, $\mathsf{prices}(\sigma(s^{\mathcal{F}}))[\sigma(s^L)] = \mathsf{prices}(s^{\mathcal{F}})[s^L]$.*

Decoupled structural symmetries induce an automorphism group over the decoupled state space:

Proposition 3 *Let Π be an FDR task, and let $\mathcal{F}$ be a star factoring. If σ is a decoupled structural symmetry of Π, then σ is an automorphism of its decoupled state space.*

Next, we show that applying a permutation to a decoupled state is equivalent to applying that permutation to each of the member states of its hypercube, as illustrated in Figure 2:

Theorem 1 *Let Π be an FDR task, and $\mathcal{F}$ be a star factoring. Let $s^{\mathcal{F}}$ be a decoupled state, and let σ be a decoupled structural symmetry. Then $\sigma(s) \in [\sigma(s^{\mathcal{F}})]$ if and only if $s \in [s^{\mathcal{F}}]$, and $cost_{\sigma(s^{\mathcal{F}})}(\sigma(s)) = cost_{s^{\mathcal{F}}}(s)$.*

Proof: Let s be a state in $[s^{\mathcal{F}}]$ with cost $c = cost_{s^{\mathcal{F}}}(s)$. Then $s = \mathsf{center}(s^{\mathcal{F}}) \cup \bigcup_{F^L \in \mathcal{F}^L} s^L$, where $s^L = s[F^L]$ for all $F^L \in \mathcal{F}^L$, and $\sum_{F^L \in \mathcal{F}^L} \mathsf{prices}(s^{\mathcal{F}})[s^L] = c$. Applying the permutation σ to s results in $\sigma(s) = \sigma(\mathsf{center}(s^{\mathcal{F}})) \cup \bigcup_{F^L \in \mathcal{F}^L} \sigma(s^L)$. By Definition 4, $\mathsf{center}(\sigma(s^{\mathcal{F}})) = \sigma(\mathsf{center}(s^{\mathcal{F}}))$, and for every leaf state $t^L \in S^L : \mathsf{prices}(\sigma(s^{\mathcal{F}}))[t^L] = \mathsf{prices}(s^{\mathcal{F}})[\sigma^{-1}(t^L)]$. Hence $\sum_{F^L \in \mathcal{F}^L} \mathsf{prices}(\sigma(s^{\mathcal{F}}))[\sigma(s^L)] = \sum_{F^L \in \mathcal{F}^L} \mathsf{prices}(s^{\mathcal{F}})[s^L] = c$, and $\sigma(s) \in [\sigma(s^{\mathcal{F}})]$ with cost c.

The same argument can be used to show that if $\sigma(s) \in [\sigma(s^{\mathcal{F}})]$ then $s \in [s^{\mathcal{F}}]$ and $cost_{\sigma(s^{\mathcal{F}})}(\sigma(s)) = cost_{s^{\mathcal{F}}}(s)$. $\square$

Finding Decoupled-State Symmetries

It is obviously infeasible to compute the automorphism group of the state space $\mathcal{T}_\Pi$ of a planning task Π directly on the state space. The symmetries on the state space must be inferred from a compact representation. Prior work introduced the *problem description graph* (PDG) of a task Π, and showed that the automorphism group of this graph induces a subgroup of $Aut(\mathcal{T}_\Pi)$ (Pochter, Zohar, and Rosenschein 2011). Later, Domshlak et al. (2012) made some modifications to the definition of the PDG, mainly to allow support of general-cost actions. Considering that the number of PDG vertices is linear in the size of Π, its automorphism group can be found efficiently using off-the-shelf tools. Our definition loosely follows that by Domshlak et al. (2012):

Definition 5 *Let Π be an FDR task. The* problem description graph *of Π, PDG(Π), is the colored digraph $\langle N, E\rangle$ with nodes N, node colors $col(n)$, and edges E:*

$$N = \{n_v \mid v \in \mathcal{V}\} \cup \bigcup_{v \in \mathcal{V}} \{n_{\langle v,d\rangle} \mid d \in \mathcal{D}(v)\} \cup \{n_a \mid a \in \mathcal{A}\},$$

$$col(n) = \begin{cases} 1 & \textit{if } n = n_{\langle v,d\rangle}, \{\langle v,d\rangle\} \in G \\ 2 + \mathsf{cost}(a) & \textit{if } n = n_a, a \in \mathcal{A} \\ 0 & \textit{otherwise} \end{cases}$$

$$E = \bigcup_{v \in \mathcal{V}} \{\langle n_v, n_{\langle v,d\rangle}\rangle \mid d \in \mathcal{D}(v)\} \cup \bigcup_{a \in \mathcal{A}} \left(E_a^{\mathsf{pre}} \cup E_a^{\mathsf{eff}}\right)$$

Where E_a^{pre} and E_a^{eff} are defined as follows:

$$E_a^{\mathsf{pre}} = \{\langle n_{\langle v,d\rangle}, n_a\rangle \mid \{\langle v,d\rangle\} \in \mathsf{pre}(a)\},$$

$$E_a^{\mathsf{eff}} = \{\langle n_a, n_{\langle v,d\rangle}\rangle \mid \{\langle v,d\rangle\} \in \mathsf{eff}(a)\}.$$

The automorphism group of PDG(Π) induces a set of structural symmetries of Π. By the same recipe we create a slightly modified version of the PDG, which induces the decoupled structural symmetries of Π:

Definition 6 *Let Π be an FDR task, let $\mathcal{F}$ be a star factoring, and let $\langle N, E\rangle$ be the PDG of Π. The* factored problem description graph *of Π given $\mathcal{F}$, PDG$(\Pi, \mathcal{F})$, is the colored digraph $\langle N', E'\rangle$ with: $N' = N \cup \{n_L \mid F^L \in \mathcal{F}^L\}$ node colors $col(n_L) = -1$; and edges $E' = E \cup \bigcup_{F^L \in \mathcal{F}^L} \{\langle n_L, n_v\rangle \mid v \in F^L\}$.*

We ensure the properties required to only obtain decoupled structural symmetries by adding the n_L nodes. By attaching each n_L node to the variables in F^L, leaf variables can only be mapped to leaf variables (so center variables can only be mapped to center variables). By coloring all n_L nodes in the *same* color, we allow permutations across leaf factors. Thus, because all variables of a particular leaf F^L are connected to exactly one n_L, we guarantee that *if* there exist symmetries that permutes a variable of F^{L_1} to one of F^{L_2}, then all variables of F^{L_1} must be mapped into variables of F^{L_2}.

Proposition 4 *Let Π be an FDR task, and let $\mathcal{F}$ be a star factoring. Every automorphism of PDG$(\Pi, \mathcal{F})$ corresponds to a decoupled structural symmetry of Π.*

Symmetry Breaking in Decoupled Search

Exploiting the symmetries of a planning task by pruning states from the search space has proved to be highly beneficial in standard search. We propose *decoupled orbit space search (DOSS)*, that applies symmetry pruning in the decoupled state space. Like in orbit space search, DOSS replaces each new decoupled state by its *canonical* representative. We next provide further details about how exactly a canonical decoupled state is computed and, once the search is completed, how a valid (optimal) plan can be reconstructed from the decoupled orbit state space.

Mapping to Canonical Representatives

In orbit space search (OSS), every state is replaced by its canonical representative. The *perfect* canonical state is defined as the one with minimal lexicographic ordering, given an arbitrary total order on variable/value pairs. Because finding the coarsest relation is **NP**-hard (Luks 1993), OSS approximates canonical representatives through a local search procedure, whose search nodes correspond to states and whose search transitions correspond to the application of one of the generators of the structural symmetries group. Search states are ranked based on their lexicographical ordering. The search stops at a local minimum.

We adapt this idea to handle decoupled states. We define the perfect canonical decoupled state to be one with a lexicographically minimal center state given an arbitrary total order on the values of center variables, and among these center-minimal decoupled states, one whose pricing function is lexicographically minimal according to an arbitrary total order on the set S^L of leaf states. (Preferring center-minimality here reduces the number of different center states among canonical decoupled states, which tends to be beneficial given the focus of decoupled search on the center.)

Like in OSS, we approximate canonical representatives via a local search procedure. For efficiency reasons, we divide this local search into two phases. A permutation σ is called *center-affecting* for a decoupled state $s^{\mathcal{F}}$, if $\sigma(\mathsf{center}(s^{\mathcal{F}})) \neq \mathsf{center}(s^{\mathcal{F}})$. Otherwise, we say that σ is *center-stable* for $s^{\mathcal{F}}$. We first perform a local search only considering center-affecting permutations, in order to obtain a decoupled state that is a local minimum with respect to the center state. Note that these center-affecting permutations may also affect the leaves, but are only applied if they improve the current center state. We then perform a second local search from that state, using only center-stable permutations. Dividing the search into two phases reduces the computational overhead to obtain the canonical state, since the more expensive checks of whether a permutation produces a lexicographically smaller pricing function are only performed for center-stable permutations.

Solution Reconstruction

When the search stops once a decoupled goal state $s^{\mathcal{F}}_G$ is found, the sequence of center actions leading to $s^{\mathcal{F}}_G$ is not guaranteed to be valid, because a permutation could have been applied in every step. Let $\sigma_0(s^{\mathcal{F}}_0) \xrightarrow{a_1} \sigma_1(s^{\mathcal{F}}_0[\![a_1]\!]) \dots \xrightarrow{a_k} \sigma_k(s^{\mathcal{F}}_{k-1}[\![a_k]\!])$ be such a center path, where $s^{\mathcal{F}}_0 = I^{\mathcal{F}}$. To get a plan for the given planning task, we need to obtain a valid center path, reconstruct the compliant leaf paths, and embed them into the center path.

We obtain a valid center path similarly to standard plan reconstruction in OSS (Domshlak, Katz, and Shleyfman 2012). This procedure retrieves all permutations $(\sigma_0, \dots, \sigma_k)$ applied during the search and then reconstructs a plan from the (real) initial decoupled state to a goal state by finding the applicable actions a'_i that produce the unpermuted state in every step. More formally, the reconstruction retrieves the following path:

$$I^{\mathcal{F}} \xrightarrow{a'_1} \sigma_0^{-1}(I^{\mathcal{F}}[\![a_1]\!]) \dots \xrightarrow{a'_k} (\sigma_0^{-1} \circ \sigma_1^{-1} \circ \dots \circ \sigma_{k-1}^{-1})(s^{\mathcal{F}}_{k-1}[\![a_k]\!])$$

Having a valid center path, we apply the standard solution construction process of decoupled search (Gnad, Hoffmann, and Domshlak 2015). This process takes low-order polynomial time in the size of the leaf state spaces and the length of the center path.

Completeness and Optimality

We prove that DOSS preserves the completeness and optimality of search algorithms giving those guarantees.

Lemma 1 *Let Π be an FDR task, let $\mathcal{F}$ be a star factoring, and let σ be a decoupled structural symmetry. Let $s^{\mathcal{F}}$ be a decoupled state. Then, $h^*(\sigma(s^{\mathcal{F}})) = h^*(s^{\mathcal{F}})$.*

Proof: By Theorem 1, for every $p \in [s^{\mathcal{F}}]$ with cost c we have $\sigma(p) \in [\sigma(s^{\mathcal{F}})]$ with the same cost. By the properties of structural symmetries, $h^*(p) = h^*(\sigma(p))$ for all $p \in [s^{\mathcal{F}}]$. Therefore, $h^*(s^{\mathcal{F}}) = \min_{p \in [s^{\mathcal{F}}]} cost_{s^{\mathcal{F}}}(p) + h^*(p) = \min_{p \in [\sigma(s^{\mathcal{F}})]} cost_{\sigma(s^{\mathcal{F}})}(p) + h^*(p) = h^*(\sigma(s^{\mathcal{F}}))$. □

Our claim now follows by an argument very similar to that for OSS (Domshlak, Katz, and Shleyfman 2015):

Theorem 2 *DOSS preserves both, completeness and optimality, of the search algorithm employed.*

Proof: Let Π be an FDR task, and let $\mathcal{F}$ be a star factoring. DOSS performs decoupled search by replacing each decoupled state $s^{\mathcal{F}}$ with its canonical representative $\sigma(s^{\mathcal{F}})$, where σ is an arbitrary decoupled structural symmetry. By Lemma 1, the goal distance of $\sigma(s^{\mathcal{F}})$ and $s^{\mathcal{F}}$ is the same (note that $h^*(s^{\mathcal{F}})$ takes into account both the center and leaf cost). Thus, an optimal plan for $s^{\mathcal{F}}$ has the same cost as an optimal plan for $\sigma(s^{\mathcal{F}})$ and we can safely replace $s^{\mathcal{F}}$ by $\sigma(s^{\mathcal{F}})$. The claim follows as decoupled search preserves completeness and optimality. □

Separation from Base Methods

We analyze the theoretical differences between DOSS, its two base components OSS and DS, and standard search which we refer to by STD. To be able to compare the methods independent of which search algorithm is used, we measure the size of the search space as the number of reachable search nodes. We assume a perfect canonical state for OSS and DOSS, though approximations are used in practice.

We say that a method A is *exponentially separated* from method B if there exists a family of planning tasks $\{\Pi_n\}$

whose search space is exponential in the size of Π_i under B, yet is polynomial under A. Previous work already proved that OSS and DS are exponentially separated from STD. Since OSS and DS are orthogonal, i. e., each can yield exponential separations over STD in different examples, it follows that they are exponentially separated from one another. By the same reasoning, DOSS is exponentially separated from each of OSS and DS. More interestingly, there exist families of planning tasks where OSS is exponentially separated from *both* its base components:

Theorem 3 *There exist families of planning tasks* $\{\Pi_n\}$ *with factoring* $\mathcal{F}_n$, *structural symmetries* Γ_n, *and decoupled structural symmetries* Γ_n^d *such that the DOSS search space has size polynomial in the size of* Π_n, *while both the OSS search space and the DS search space have size exponential in the size of* Π_n.

Proof Sketch: It suffices to construct a planning task Π_n with n symmetric variables in the center and n leaves that are not symmetric. The search space of OSS is exponential in n because it has to consider all combinations of leaf states. The search space of DS is exponential in n because it enumerates all combinations for the n center variables. However, DOSS has a polynomial number of center states and pricing functions. □

Note that the exponential separation does not imply that the search space under DOSS will always be smaller than that of its components. While this is true for OSS compared to STD, DS does not dominate STD. Indeed, there exist examples where STD (and thus also OSS) has a search space exponentially smaller than that of DS (Gnad and Hoffmann 2015). The same examples can be used to show that STD and OSS are exponentially separated from DOSS. However, as we shall see next, in practice the search space of DOSS does tend to be smaller than that of DS and OSS.

Experiments

We implemented our techniques in Fast Downward (Helmert 2006), extending Gnad et al.'s (2015) implementation of DS and the symmetries implementation of Metis (Alkhazraji et al. 2014). To obtain the decoupled structural symmetries of a planning task Π, we use the BLISS tool (Junttila and Kaski 2007) on the factored problem description graph of Π. To obtain star factorings, we use Gnad et al.'s *X-shape* factoring strategy, which greedily computes a factoring that maximizes the number of leaf factors. Like Gnad et al., if the obtained factoring has less than two leaf factors, we *abstain* from solving the task (the rationale being that decoupled search addresses conditional dependencies across *multiple* leaves).

We conduct experiments in optimal planning, in satisficing planning, as well as in proving unsolvability. In all of these settings, we use all IPC STRIPS benchmarks (1998 – 2016) where the factoring method does not abstain. The experiments were performed on a cluster of Intel E5-2660 machines running at 2.20 GHz, with time (memory) cut-offs of 30 minutes (4 GB).

		Blind Search				LM-cut			
Domain	#	STD	OSS	DS	DOSS	STD	OSS	DS	DOSS
Childsnack	20	0	**6**	0	**6**	0	**6**	0	**6**
Depots	22	4	**6**	4	5	7	8	7	**9**
Driverlog	20	7	7	**11**	**11**	13	13	13	13
Elevators08	30	14	**15**	9	11	22	22	**23**	**23**
Elevators11	20	12	**13**	7	9	18	18	18	18
Floortile11	20	2	2	2	2	7	**8**	7	**8**
Floortile14	20	0	0	0	0	6	**8**	6	**8**
Logistics00	28	10	12	**22**	**22**	20	20	**28**	**28**
Logistics98	35	2	3	**4**	**4**	6	6	6	**8**
Miconic	145	50	**51**	36	38	136	**137**	135	136
NoMystery	20	8	9	**17**	**17**	14	15	**20**	**20**
Pathways	30	4	4	4	4	**5**	**5**	4	4
Rovers	40	6	6	**7**	**7**	7	7	**9**	**9**
Satellite	36	6	6	6	6	7	**13**	7	**13**
TPP	29	5	6	**23**	**23**	5	7	18	**20**
Transport11	20	6	6	6	6	6	**7**	6	6
Transport14	20	**7**	**7**	4	4	6	6	6	6
Woodwork08	24	6	6	**7**	**7**	12	14	17	**18**
Woodwork11	15	2	2	**3**	**3**	8	9	11	**12**
Zenotravel	20	8	8	**11**	**11**	13	13	13	13
Others	37	14	14	14	14	14	14	14	14
$\sum$	651	173	189	197	**210**	332	356	368	**392**

Table 1: Coverage on all optimal planning benchmarks that have an X-shape factoring. All configurations use A* search. Domains with equal coverage in all configurations are summarized in "Others". Best results marked in **bold face**.

Optimal Planning

Most prior work using symmetry breaking in classical planning focused on its application in optimal planning, where reducing the size of the search space by pruning states is highly beneficial (e. g. Pochter, Zohar, and Rosenschein (2011), Domshlak, Katz, and Shleyfman (2012)). Here, we show that the same holds when applying symmetries to the decoupled state space.

Table 1 shows the number of solved instances by the baseline algorithms – standard search (STD), orbit space search (OSS), and decoupled search (DS) – and our new decoupled orbit space search (DOSS). DOSS clearly excels in terms of total coverage. Taking a more detailed look at the individual domains reveals that, when using blind search, DOSS dominates DS in all domains, being strictly better in five. The comparison to OSS illustrates the complementarity of both approaches. Although the coverage increases in domains in which DS does not perform well, enabling symmetry pruning cannot completely compensate for the lower coverage of DS in, e. g., Elevators, Miconic, or Transport.

Matters are different when using LM-cut (Helmert and Domshlak 2009). DOSS almost always inherits the strength of its best component method. There are only three benchmark instances solved by either of OSS or DS, but not by DOSS. Even more impressively, *in five domains DOSS strictly dominates both of its components, and in two of these domains by more than one instance*. This shows that true synergies, DOSS being more than the sum of its components, occur not only in theory (cf. Theorem 3), but also in practice.

In Figure 3, the four left-most scatter plots (the two

		no preferred operators				preferred operators			
Domain	#	STD	OSS	DS	DOSS	STD	OSS	DS	DOSS
Childsnack	20	0	**4**	0	**4**	3	14	6	**20**
Depots	22	14	17	19	**21**	18	19	20	**22**
Driverlog	20	18	18	**20**	**20**	20	20	20	20
Elevators11	20	18	18	**20**	**20**	20	20	20	20
Floortile11	20	6	6	4	**7**	6	**7**	6	**7**
Logistics98	35	26	31	**35**	**35**	35	35	35	35
Mystery	4	0	0	**1**	0	**1**	**1**	**1**	0
NoMystery	20	9	10	**19**	**19**	10	11	**19**	**19**
Pathways	30	11	11	**13**	**13**	20	20	20	20
Rovers	40	**23**	**23**	22	22	40	40	40	40
Satellite	36	30	**35**	33	**35**	36	36	36	36
TPP	29	21	18	25	**26**	29	29	29	29
Transport08	30	16	17	**30**	**30**	28	27	**30**	**30**
Transport11	20	0	1	**20**	**20**	11	10	**20**	**20**
Transport14	20	0	0	**20**	**20**	6	7	**20**	**20**
Woodwork11	20	19	19	**20**	**20**	20	20	20	20
Others	274	256	256	256	256	256	256	256	256
$\sum$	660	467	484	557	**568**	559	572	598	**614**

Table 2: Coverage on all satisficing planning benchmarks that have an X-shape factoring. All configurations use greedy best-first search with the h^{FF} heuristic. Domains with equal coverage in all configurations are summarized in "Others". Best results marked in **bold face**.

columns marked by "OPT") shed further light on the relation between DOSS and its components when using LM-cut. The upper plots show the number of expanded states until the last f-layer in A*. In both plots, the advantage of DOSS over OSS and DS is pronounced, even more so for OSS. The bottom plots show that this reduction in search space size nicely translates into reduced runtime. Especially interesting is that DOSS seems to come at a very low risk: the computational overhead pays off, most of the time.

Satisficing Planning

We next consider satisficing planning. We run greedy best-first search using the h^{FF} heuristic (Hoffmann and Nebel 2001), both with and without preferred operator pruning. It turns out to be beneficial, in this setting, to apply only the center-affecting permutations in the quest for canonical representatives: applying the center-stable automorphisms typically incurs more runtime overhead than search benefit.

Table 2 shows coverage results of DOSS compared to the baseline algorithms. Independent of the usage of preferred operators, DOSS very consistently beats its component methods. There are only two domains where DOSS loses coverage. In Mystery, due to the computational overhead of obtaining the symmetry relation on the very large leaf state spaces in this domain. And in Rovers, due to the worse performance of DS with respect to STD. Except for those cases, DOSS again inherits the strengths of both DS and OSS. And again, it sometimes surpasses those strengths – in Depots and Childsnack – exhibiting true synergy. Indeed, in Childsnack with preferred operators, DOSS outclasses both its components, increasing coverage by $+6$ relative to OSS and by $+14$ relative to DS.

In Figure 3, the plots in the two "SAT" columns show number of expanded states (top) and runtime (bottom) when not using preferred operators. The comparison of DOSS to OSS is extremely favorable, showing in particular that using decoupled search on top of symmetry breaking incurs hardly any risk. In the comparison of DOSS to DS, the improvements are smaller but still significant. In the runtime plot, observe that, the more difficult a task is, the bigger is the benefit of applying symmetries: for tasks that are quickly solved by both algorithms, DS can be significantly faster, yet on more challenging tasks the overhead of symmetry breaking is typically outweighed by the reduced search space size.

Proving Unsolvability

Proving unsolvability has recently gained attention (Bäckström, Jonsson, and Ståhlberg 2013; Hoffmann, Kissmann, and Torralba 2014), culminating in the Unsolvability IPC 2016. To prove a task unsolvable, one possibility is to completely exhaust the reachable state space. Since both decoupled search and symmetry breaking are techniques developed to reduce the size of the state space, we apply DOSS to exhaust state spaces. We do so on the domains of the Unsolvability IPC'16; we also run all standard IPC benchmarks (of both, the optimal and satisficing tracks), which of course are not unsolvable, but which provide an additional evaluation of the ability (or lack thereof) to exhaust state spaces.

The results in Table 3 shed light on the number of state spaces each technique can entirely build. For readability, we aggregate different versions of each domain into a single row. We compare standard search and orbit space search against two variants of decoupled search and DOSS, called OPT and COM, of which OPT preserves completeness and optimality while COM preserves completeness only. OPT is exactly what we described above; COM simplifies this by maintaining not pricing functions, but *reachability functions*

				DS		DOSS	
Domain	#	STD	OSS	OPT	COM	OPT	COM
Childsnack	40	0	6	0	0	6	**12**
Depots	22	4	5	3	5	4	**8**
Driverlog	20	5	7	8	**10**	8	**10**
Elevators	100	21	28	8	41	10	**42**
Logistics	63	12	14	23	**25**	24	**25**
Miconic	145	45	51	30	**145**	35	137
NoMystery	40	11	12	25	**28**	26	**28**
Satellite	36	4	**5**	4	4	**5**	**5**
TPP	29	5	6	11	11	14	**16**
Transport	140	25	29	18	**34**	20	**34**
Woodworking	87	11	12	16	16	16	**17**
Others	177	21	21	21	21	21	21
Unsolvability IPC'16							
BagTransport	29	7	7	4	**11**	5	**11**
NoMystery	24	2	2	**12**	**12**	**12**	**12**
Rovers	20	7	7	8	8	**9**	**9**
$\sum$	972	180	212	191	371	215	**387**

Table 3: Number of tasks for which the reachable part of the state space could be built (top) or that could be proved unsolvable (bottom). OPT/COM: Decoupled search variants that preserve optimality/completeness. Domains with equal numbers in all configurations are summarized in "Others". Best results marked in **bold face**.

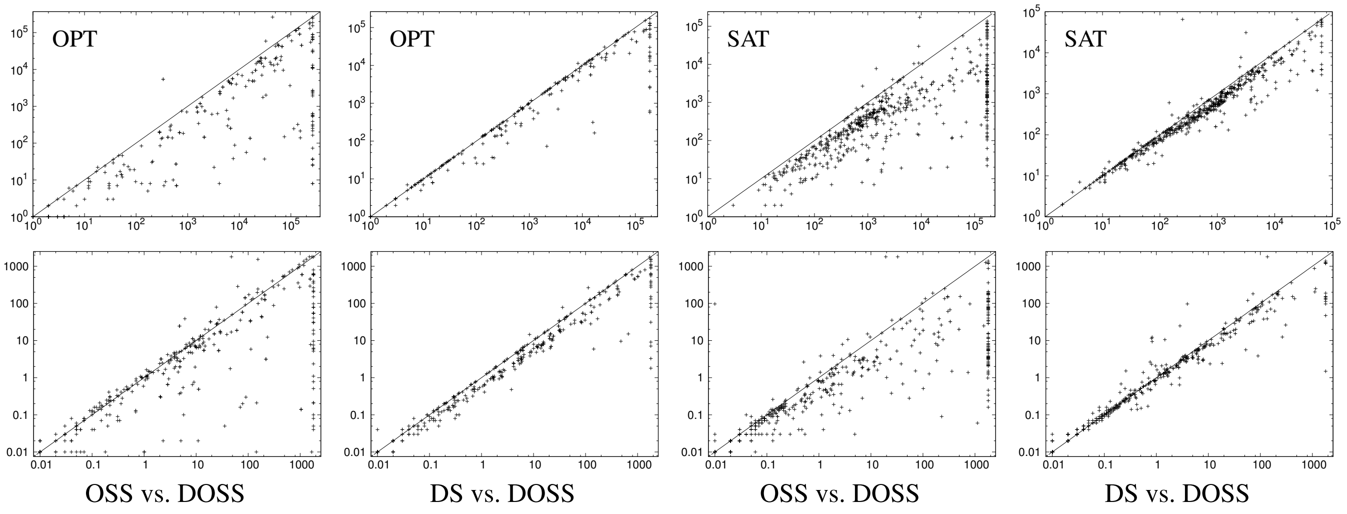

Figure 3: Scatter plots, with a data point per instance, showing the number of expanded nodes (top) (until last f-layer for OPT configurations) and runtime in seconds (bottom) of algorithms "X vs. Y", with X on the x-axis and Y on the y-axis. OPT configurations use A* with the LM-cut heuristic, SAT use greedy best-first search with the h^{FF} heuristic.

which merely indicate, for each leaf state s^L, whether or not s^L has been reached yet. (Reachability functions are equivalent to pricing functions in a modified task where all action costs are assumed to be 0.) Clearly, COM suffices to prove unsolvability, so it is our main focus here; results for OPT are included for reference.

According to Table 3, the OPT variant of DS is highly complementary to OSS, and enabling symmetry pruning on top of DS does not win back the advantage of orbit space search in most domains where OSS is better. Compared to DS, performance is still improved significantly.

The results for the more suitable COM variant, however, are much better. The COM variant of DOSS dominates both baselines in all domains but Miconic, where DOSS explores larger parts of the state space than DS.[1] In Childsnack, Depots, Elevators, TPP, Woodworking, and UIPC'16 Rovers, DOSS outperforms both its components, exhibiting once again a strong practical ability for exploiting synergies.

Figure 4 shows more detailed results in terms of state space size (top) and runtime (bottom), using the more appropriate COM variant of decoupled search. The general picture is very similar to that of optimal and satisficing planning. Compared to OSS, the advantage of decoupled search yields huge search space and runtime reductions, at a very low risk. Compared to DS, the improvements are smaller but still significant. It is noteworthy that almost all bad cases for DS vs. DOSS, i. e., points above the diagonal, in particular the cluster around $x = 10^3$ in the state space size plot, are due to the Miconic domain. In all other domains, the runtime of DOSS merely increases by a small constant factor, and the state space size is mostly smaller than that of DS.

[1]This is counter-intuitive because symmetries reduce the size of the decoupled state space. However, using symmetries may reduce the power of the dominance pruning that decoupled search uses in order to prune states with higher prices.

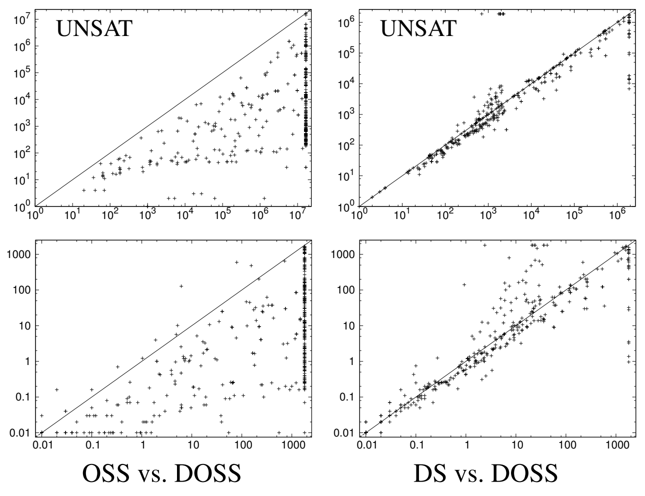

Figure 4: Like Figure 3, for COM state space exhaustion.

Conclusion

Symmetry breaking and star-topology decoupled search are complementary search reduction methods. As we show, they can be combined. That combination may be stronger than both components, exhibiting true synergy. Most remarkably, such synergy is not only possible in theory, but happens quite regularly on the standard planning benchmarks.

Future work includes the further exploration of algorithm space – with star-topology decoupled search being a reformulation of the search itself, essentially all search reduction methods for forward search are potentially applicable (Gnad, Wehrle, and Hoffmann 2016; Torralba et al. 2016). Combinations of several methods, like decoupled search + partial-order reduction + symmetries, are interesting as well. Beyond this, another exciting direction is the application of these ideas beyond planning, in particular in verification where state space exhaustion is paramount.

Acknowledgments

D. Gnad was supported by the German Research Foundation (DFG), under grant HO 2169/6-1. Á. Torralba was supported by the German Federal Ministry of Education and Research (BMBF) through CISPA, grant no. 16KIS0656. A. Shleyfman was supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

References

Alkhazraji, Y.; Katz, M.; Mattmüller, R.; Pommerening, F.; Shleyfman, A.; and Wehrle, M. 2014. Metis: Arming fast downward with pruning and incremental computation. In *IPC 2014 planner abstracts*, 88–92.

Amir, E., and Engelhardt, B. 2003. Factored planning. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 929–935.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Computational Intelligence* 11(4):625–655.

Bäckström, C.; Jonsson, P.; and Ståhlberg, S. 2013. Fast detection of unsolvable planning instances using local consistency. In *Proc. of the Symposium on Combinatorial Search (SOCS)*, 29–37.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):279–298.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *Proc. of the National Conference of the American Association for Artificial Intelligence (AAAI)*, 809–814.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 28–35.

Brafman, R., and Domshlak, C. 2013. On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence* 198:52–71.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2015. Symmetry breaking in deterministic planning as forward search: Orbit space search algorithm. *Technical Report IS/IE-2015-02*.

Emerson, E. A., and Sistla, A. P. 1996. Symmetry and model-checking. *Formal Methods in System Design* 9(1/2):105–131.

Fabre, E.; Jezequel, L.; Haslum, P.; and Thiébaux, S. 2010. Cost-optimal factored planning: Promises and pitfalls. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 65–72.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 956–961.

Gnad, D., and Hoffmann, J. 2015. Beating LM-cut with h^{max} (sometimes): Fork-decoupled state space search. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry breaking in star-topology decoupled search. Technical report, Saarland University. Available at `http://fai.cs.uni-saarland.de/gnad/papers/icaps17-tr.pdf`.

Gnad, D.; Hoffmann, J.; and Domshlak, C. 2015. From fork decoupling to star-topology decoupling. In *Proc. of the Symposium on Combinatorial Search (SOCS)*.

Gnad, D.; Wehrle, M.; and Hoffmann, J. 2016. Decoupled strong stubborn sets. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 162–169.

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

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Kissmann, P.; and Torralba, Á. 2014. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In *Proc. of the European Conference on Artificial Intelligence (ECAI)*.

Junttila, T., and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc of. the Workshop on Algorithm Engineering and Experiments (ALENEX)*, 135–149.

Kelareva, E.; Buffet, O.; Huang, J.; and Thiébaux, S. 2007. Factored planning using decomposition trees. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1942–1947.

Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

Long, D., and Fox, M. 1999. Efficient implementation of the plan graph in stan. *Journal of Artificial Intelligence Research* 10:87–115.

Luks, E. M. 1993. Permutation groups and polynomial-time computation. In *Groups and Computation, DIMACS Series in Disc. Math. and Th. Comp. Sci.*, volume 11, 139–175.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In *Proc. of the National Conference of the American Association for Artificial Intelligence (AAAI)*.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 32–41.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and symmetries in classical planning. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, 3371–3377.

Sievers, S.; Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Factored symmetries for merge-and-shrink abstractions. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, 3378–3385.

Starke, P. 1991. Reachability analysis of petri nets using symmetries. *Journal of Mathematical Modelling and Simulation in Systems Analysis* 8(4/5):293–304.

Torralba, Á.; Gnad, D.; Dubbert, P.; and Hoffmann, J. 2016. On state-dominance criteria in fork-decoupled search. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Integrating partial order reduction and symmetry elimination for cost-optimal classical planning. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.