Journal of Artificial Intelligence Research 65 (2019) 343-392 Submitted 12/2018; published 07/2019

Strong Stubborn Set Pruning for Star-Topology Decoupled State Space Search

Daniel Gnad GNAD@CS.UNI-SAARLAND.DE
Jörg Hoffmann HOFFMANN@CS.UNI-SAARLAND.DE
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany

Martin Wehrle MARTIN.WEHRLE79@GMAIL.COM
University of Basel
Basel, Switzerland

Abstract

Analyzing reachability in large discrete transition systems is an important sub-problem in several areas of AI, and of CS in general. State space search is a basic method for conducting such an analysis. A wealth of techniques have been proposed to reduce the search space without affecting the existence of (optimal) solution paths. In particular, *strong stubborn set (SSS)* pruning is a prominent such method, analyzing action dependencies to prune commutative parts of the search space. We herein show how to apply this idea to *star-topology decoupled state space search*, a recent search reformulation method invented in the context of classical AI planning.

Star-topology decoupled state space search, short *decoupled search*, addresses planning tasks where a single *center* component interacts with several *leaf* components. The search exploits a form of conditional independence arising in this setting: given a fixed path π^C of transitions by the center, the possible leaf moves *compliant* with π^C are independent across the leaves. Decoupled search thus searches over center paths only, maintaining the compliant paths for each leaf separately. This avoids the enumeration of combined states across leaves.

Just like standard search, decoupled search is adversely affected by commutative parts of its search space. The adaptation of strong stubborn set pruning is challenging due to the more complex structure of the search space, and the resulting ways in which action dependencies may affect the search. We spell out how to address this challenge, designing optimality-preserving *decoupled strong stubborn set (DSSS)* pruning methods. We introduce a design for star topologies in full generality, as well as simpler design variants for the practically relevant *fork* and *inverted fork* special cases. We show that there are cases where DSSS pruning is exponentially more effective than both, decoupled search and SSS pruning, exhibiting true synergy where the whole is more than the sum of its parts. Empirically, DSSS pruning reliably inherits the best of its components, and sometimes outperforms both.

1. Introduction

Analyzing reachability in large discrete transition systems is an important sub-problem in several areas of AI, and of CS in general. In particular, this pertains to classical AI Planning, which we focus on here. Given an initial state, a goal condition, and a set of deterministic actions, all described relative to a set of finite-domain state variables, the planning task is to find a sequence of actions leading from the initial state to a state that satisfies the goal condition. In other words, we aim to

analyze the reachability of the goal condition starting from the initial state, in the large deterministic transition system given by the task's state space.

Forward state space search, starting at the initial state and exploring the state space until a goal state is reached, is a basic method for conducting reachability analysis. For *satisficing planning*, where the objective is to find any (not necessarily optimal) solution path, today the most successful means to make forward search effective is *heuristic search*, guiding the search greedily towards the goal (e.g., McDermott, 1999; Bonet & Geffner, 2001; Hoffmann & Nebel, 2001; Gerevini, Saetti, & Serina, 2003; Helmert, 2006; Richter & Westphal, 2010; Domshlak, Hoffmann, & Katz, 2015). For *optimal planning*, where optimality must be guaranteed, heuristic search – using the well-known A* algorithm (Pearl, 1984) – also is one of the most successful methods (e.g., Edelkamp, 2001; Helmert & Domshlak, 2009; Helmert, Haslum, Hoffmann, & Nissim, 2014). However, it is known that even "almost perfect" heuristic functions (Gaschnig, 1977) – underestimating goal distance by at most an additive constant – do not prevent A* from exponential search, even on simple standard toy planning benchmarks (Helmert & Röger, 2008). This observation motivates the need for other, complementary, ways of enhancing forward search.

A wealth of such techniques have been developed, including symbolic representations (e.g., Bryant, 1986; Edelkamp & Helmert, 1999; Torralba, Alcázar, Kissmann, & Edelkamp, 2017), symmetry breaking (e.g., Starke, 1991; Pochter, Zohar, & Rosenschein, 2011; Domshlak, Katz, & Shleyfman, 2012), dominance pruning (e.g., Hall, Cohen, Burkett, & Klein, 2013; Torralba & Hoffmann, 2015), and partial-order reduction (e.g., Valmari, 1989; Godefroid & Wolper, 1991; Edelkamp, Leue, & Lluch-Lafuente, 2004; Alkhazraji, Wehrle, Mattmüller, & Helmert, 2012; Wehrle, Helmert, Alkhazraji, & Mattmüller, 2013; Wehrle & Helmert, 2014). We herein consider the latter, specifically *strong stubborn set (SSS)* pruning, which was first devised in Verification (Valmari, 1989), and was later adapted to AI Planning (Alkhazraji et al., 2012; Wehrle et al., 2013; Wehrle & Helmert, 2014). An SSS for a state s is a subset $\mathcal{A}_s$ of applicable actions guaranteed to contain the starting action of at least one optimal solution for s. In a nutshell, such a set $\mathcal{A}_s$ is derived by selecting an open part p of the goal condition, collecting all actions a that may recursively be used to *enable* p, and collecting all actions a' that these a *interfere* with. Actions not included by this process can be safely ignored in s, as they are not relevant to p, and as they are commutative with $\mathcal{A}_s$, i.e., they can still be applied later on if needed for some other part of the goal.

Our contribution consists in showing how to apply this idea in *star-topology decoupled state space search*, short *decoupled search*, a search decomposition method recently invented in AI Planning (Gnad & Hoffmann, 2015; Gnad, Hoffmann, & Domshlak, 2015; Gnad & Hoffmann, 2018).

Decoupled search is a form of *factored planning* (Sacerdoti, 1974; Knoblock, 1994; Lansky & Getoor, 1995; Amir & Engelhardt, 2003; Brafman & Domshlak, 2006; Kelareva, Buffet, Huang, & Thiébaux, 2007; Brafman & Domshlak, 2008; Fabre, Jezequel, Haslum, & Thiébaux, 2010; Nissim, Brafman, & Domshlak, 2010; Crosby, Rovatsos, & Petrick, 2013; Brafman & Domshlak, 2013; Nissim & Brafman, 2014; Wang & Williams, 2015), whose basic idea is to view the planning task as a set of interacting components – the *factors*, characterized by disjoint subsets of state variables – and to organize the search in terms of these components. Individual approaches to factored planning differ widely in how that is done. A rough classification of prior work is into hierarchical factored planning, where factors are used in an abstraction/refinement hierarchy, and localized factored planning, where local per-factor planning is combined with global constraint resolution. Decoupled search can be viewed as a form of localized factored planning, unique in the

assumption and exploitation of a particular structural profile of cross-factor interactions, namely a *star topology*, which lends itself to a specialized form of global constraint resolution.

In a star topology, a single *center* factor interacts directly with several *leaf* factors, but the leaves interact with each other only indirectly via the center. As prior work on decoupled search (Gnad & Hoffmann, 2015; Gnad et al., 2015; Gnad & Hoffmann, 2018) has shown, such topologies can be identified automatically in a pre-process to planning, based on the task's *causal graph* (Knoblock, 1994; Jonsson & Bäckström, 1995; Brafman & Domshlak, 2003; Helmert, 2006).

The key advantage of star topologies is a particular form of "conditional independence": given a fixed path π^C of transitions by the center, the possible leaf moves *compliant* with π^C are independent across the leaves. Decoupled search exploits this property by searching over center paths π^C only, maintaining the compliant paths for each leaf separately. This avoids the enumeration of combined states across leaves, and dramatically reduces the state space on planning benchmarks where a pronounced star topology (many leaves) can be identified. In catchy (though imprecise) analogy to conditional independence in graphical models, star-topology decoupling "instantiates" the center to break the dependencies between the leaves.

Despite this prowess, decoupled search remains a state space search approach, and can suffer from the same kinds of deficiencies. All of the abovementioned enhancements – heuristic search, symbolic representations, symmetry breaking, dominance pruning, partial-order reduction – remain relevant in principle. The question is whether these methods are applicable to decoupled search, and if so, how. The initial work on decoupled search (Gnad & Hoffmann, 2015; Gnad et al., 2015; Gnad & Hoffmann, 2018) already answered this question for heuristic search, and subsequent work has provided an answer for dominance pruning (Torralba, Gnad, Dubbert, & Hoffmann, 2016), and symmetry breaking (Gnad, Torralba, Shleyfman, & Hoffmann, 2017c). Herein we provide an answer for partial-order reduction using strong stubborn set pruning.

The major challenge in doing so is the more complex structure of the search space. Each *decoupled state* $s^{\mathcal{F}}$ in the search consists of a center path π^C along with, for each leaf factor, the set of leaf states s^L *reached* on π^C-compliant paths. The interpretation of $s^{\mathcal{F}}$ is that any one of the reached s^L *can* be committed to if needed. But this commitment is postponed to plan extraction time, when every leaf has a reached goal leaf state $s^L_\star$, and a global plan is obtained by augmenting π^C, for every leaf factor, with a cheapest π^C-compliant leaf path ending in that $s^L_\star$.

Given this search structure, for the leaf factors, the distinction between "past" (search path so far) and "future" (remaining search path) becomes complicated, because even the path the leaf takes up to $s^{\mathcal{F}}$ will be committed to only in the future. In particular, even if a leaf state s^L is flagged as being reached in $s^{\mathcal{F}}$, SSS pruning needs to reason about the enablers of s^L, i.e., about the actions that *might* be committed to in order to support s^L at plan extraction time.

We analyze these issues in detail, and spell out how to identify optimality-preserving *decoupled strong stubborn sets (DSSS)*. We begin with the general case, designing DSSS for arbitrary star topologies. Having to handle all possible cases, the construction turns out to be rather inclusive, selecting potential enablers of reached leaf states when starting from essentially any other reached leaf state. Hence, we subsequently investigate easier but still practically relevant special cases, allowing a more selective choice of enablers. We consider *fork* topologies, where the leaves depend on the center but not vice versa; *inverted-fork* topologies, where the dependencies are inverse; as well as fork and inverted-fork leaves occuring as part of general star topologies.[1]

1. The names "fork" and "inverted fork" are due to work by Katz and Domshlak (2008, 2010) in a different context.

We evaluate our DSSS techniques from both a theoretical and a practical perspective. For the theoretical part, we analyze *exponential separations*: example task families scaling in a size parameter n, exponentially separating some method A from another one B in that A yields search space size polynomial in n while B yields search space size exponential in n. We show that decoupled search and SSS pruning are exponentially separated from each other, and that DSSS pruning is exponentially separated from each of them. Beyond the latter, we show that there are example task families – not complex artificial examples, but simple variants of the Logistics benchmark – exhibiting true synergy where the whole is strictly more than the sum of its parts: cases where the exponential separation is relative to *both* decoupled search and SSS, i.e., search space size remains exponential under each of these methods, yet becomes polynomial under DSSS pruning.

For the practical evaluation, we run our techniques on standard benchmarks from the International Planning Competition (IPC). We consider optimal planning as well as proving unsolvability. DSSS pruning reliably inherits the best of decoupled search and SSS pruning. Sometimes – being more than the sum of its parts – it outperforms both. Where both techniques are effective, DSSS pruning is able to compete with state-of-the-art systems in both optimal planning, and proving unsolvability, outperforming it in several domains.

Section 2 gives our planning framework and notations. Section 3 gives summary presentations of the two base methods – decoupled search and SSS pruning – that we build upon. Section 4 introduces our DSSS construction for general star topologies, Section 5 introduces the simpler constructions specialized for forks and inverted forks. Sections 6 and 7 provide the theoretical and practical evaluations respectively. We conclude in Section 8. Some technical details are only summarized in the main text. The full details are available in Appendix A.[2]

2. Preliminaries

We employ a finite-domain state variable formalization of planning (compare, e.g., Bäckström & Nebel, 1995; Edelkamp & Helmert, 1999; Helmert, 2006, 2009). A **finite-domain representation** planning task, short FDR task, is a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$. $\mathcal{V}$ is a set of **state variables**, short **variables**. Each $v \in \mathcal{V}$ is associated with its finite domain $\mathcal{D}(v)$. We identify (partial) variable assignments with sets of variable/value pairs. A complete assignment to $\mathcal{V}$ is a **state**. s_0 is the **initial state**. The **goal** $s_\star$ is a partial assignment to $\mathcal{V}$. $\mathcal{A}$ is a finite set of **actions**. Each action $a \in \mathcal{A}$ is a triple $\langle pre(a), \mathit{eff}(a), cost(a) \rangle$ where the **precondition** $pre(a)$ and **effect** $\mathit{eff}(a)$ are partial assignments to $\mathcal{V}$, and $cost(a) \in \mathbb{R}^{0+}$ is a's non-negative **cost**. We assume that $\mathit{eff}(a) \neq \emptyset$, i.e., every action *affects* at least one state variable; otherwise, a can obviously be removed. (This assumption avoids some awkward special cases from decoupled search terminology, namely actions that affect neither the center nor a leaf of a star topology.)

Given a partial assignment p, by $vars(p) \subseteq \mathcal{V}$ we denote the subset of state variables on which p is defined. For a variable subset $V \subseteq \mathcal{V}$, by $p[V]$ we denote the restriction of p to $vars(p) \cap V$. Given partial assignments p and q where $vars(p) \supseteq vars(q)$, we say that p **satisfies** q, written $p \models q$, if $p[vars(q)] = q$.

The outcome of applying an action a in state s, denoted $s[\![a]\!]$, is a state where the assignment to $vars(\mathit{eff}(a))$ is overwritten with $\mathit{eff}(a)$, leaving s unchanged elsewhere. The outcome state of an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$ is $s[\![\pi]\!] := (((s[\![a_1]\!])[\![a_2]\!]) \ldots)[\![a_n]\!]$. An action a is **applicable**

2. The present paper extends work published at IJCAI'16 (Gnad, Wehrle, & Hoffmann, 2016). That work considered fork topologies only.

to a state s if $s \models pre(a)$. An action sequence $\pi = \langle a_1, \ldots, a_n \rangle$ is applicable to s if each a_i is applicable in $s[\![\langle a_1, \ldots, a_{i-1} \rangle]\!]$.

A **plan** for state s is an applicable action sequence π s.t. $s[\![\pi]\!] \models s_\star$. The **cost** of an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$, denoted $cost(\pi)$, is the summed-up cost $\sum_{i=1}^{n} cost(a_i)$ of its actions. A plan for s_0 is called a plan for Π. A plan π for s is **optimal** if $cost(\pi)$ is minimal among all plans for s. The plan is **strongly optimal** if it is optimal and contains the minimum number of 0-cost actions among all optimal plans for s. We will need the notion of strong optimality to ensure that the search makes progress and does not get trapped in 0-cost cycles.

We will often talk about compatible vs. incompatible partial assignments (action preconditions or effects), so we introduce shorthands for this. Given partial assignments p and q, we say that p and q are **compatible**, written $p \parallel q$, if there exists no $v \in \mathcal{V}$ such that $v \in vars(p) \cap vars(q)$ and $p[v] \neq q[v]$; we say that p and q are **incompatible**, written $p \nparallel q$, if such v does exist.

A standard means to capture the structure of planning tasks is the **causal graph** (Knoblock, 1994; Jonsson & Bäckström, 1995; Brafman & Domshlak, 2003; Helmert, 2006). This is a directed graph whose vertices are the variables $\mathcal{V}$, and with an arc (u, v) if either (i) there exists $a \in \mathcal{A}$ so that $u \in vars(pre(a))$ and $v \in vars(\mathit{eff}(a))$, or (ii) there exists $a \in \mathcal{A}$ so that $u \in vars(\mathit{eff}(a))$ and $v \in vars(\mathit{eff}(a))$. Observe that such arcs capture a dependency of v on u, in the sense that either (i) to move v, we may have to move u first, or (ii) moving v may, as a side effect, move u as well.

3. Base Methods

We give summary presentations of decoupled search, and of strong stubborn sets pruning, sufficient to understand their workings and introducing the notations we will use in our analysis later on. Assume an FDR task $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$.

3.1 Decoupled Search

A **factoring** $\mathcal{F}$ is a partition of $\mathcal{V}$ into non-empty subsets F, called **factors**. $\mathcal{F}$ is a **star factoring** if $|\mathcal{F}| > 1$ and there exists $F^C \in \mathcal{F}$ such that, for every action a where $vars(\mathit{eff}(a)) \cap F^C = \emptyset$, there exists $F \in \mathcal{F}$ with $vars(\mathit{eff}(a)) \subseteq F$ and $vars(pre(a)) \subseteq F \cup F^C$. F^C is the **center** of $\mathcal{F}$, and all other factors $F^L \in \mathcal{F}^L := \mathcal{F} \setminus \{F^C\}$ are **leaves**. That is, in a star factoring, actions affecting the center are unrestricted, yet actions not affecting the center must affect exactly one leaf and have preconditions only on the center and that leaf. This is the most general kind of structure to which decoupled search is applicable.

Relevant special cases are **fork** factorings and **inverted-fork** factorings. These are defined by viewing $\mathcal{F}$ as an equivalence relation over the state variables, and restricting the structure of the causal graph's quotient graph given this relation, merging all variables of a factor into a single node. A **fork leaf** is a leaf factor F^L whose only arc in the quotient graph is (F^C, F^L), i.e., the leaf has preconditions on the center but not vice versa, and there are no actions affecting both together. An **inverted-fork leaf** is an F^L where the dependency direction is inverse, i.e., whose only arc in the quotient graph is (F^L, F^C). A fork factoring is one all of whose leaves are fork leaves, an inverted-fork factoring is one all of whose leaves are inverted-fork leaves.

Example 1. *As running examples, we will consider simple transportation tasks similar to several IPC benchmarks along these lines. Say we have two locations A and B, a single truck whose*

position is modeled by variable t with $\mathcal{D}(t) = \{A, B\}$, and n packages whose position is modeled by variables p_i with $\mathcal{D}(p_i) = \{A, B, T\}$. The truck and all packages are initially at A, and the goal is for the packages to be at B. The actions take the form drive(x, y), load(p_i, t, x), and unload(p_i, t, x) where $x, y \in \{A, B\}$. Actions costs are unit 1 for simplicity.

In what we will call the Vanilla *variant of this example, the action preconditions and effects are the commonly used ones, namely: drive(x, y) has precondition $\{t = x\}$ and effect $\{t = y\}$; load(p_i, t, x) has precondition $\{p_i = x, t = x\}$ and effect $\{p_i = T\}$; and unload(p_i, t, x) has precondition $\{p_i = T, t = x\}$ and effect $\{p_i = x\}$.*

In what we will call the NoEmpty *variant of this example, the actions are the same except that driving the truck now takes the form drive(x, y, p_i), with precondition $\{t = x, p_i = T\}$ and effect $\{t = y\}$. That is, here the truck cannot drive without having a package inside.*

Unless stated otherwise, we will consider the default factoring $\mathcal{F}$ *setting $\{t\}$ as the center, and setting each $\{p_i\}$ as a leaf. For the Vanilla example, this is a fork factoring. For the NoEmpty example, where the dependencies between t and p_i go in both directions, this is (not a fork but) a star factoring.*

We will sometimes consider the inverse factoring*, setting $\{p_1, \ldots, p_n\}$ as the center and $\{t\}$ as the (single) leaf. Observe that, for the Vanilla example, this is an inverted-fork factoring.*

Previous work on decoupled search has employed fork and inverted-fork factorings automatically found through simple and practically highly efficient **factoring strategies** based on causal graph analysis (Gnad & Hoffmann, 2015; Gnad et al., 2015). Later, more sophisticated factoring methods were proposed, that are able to detect so-called **strict-star factorings**, star factorings where any interaction between leaves is prohibited (Gnad, Poser, & Hoffmann, 2017a). We will be using the fork and inverted-fork, as well as the incident-arcs-based factoring strategies from Gnad et al. (2017a) in our experiments. The latter greedily moves variables densely connected in the causal graph into the center factor, making each weakly connected component in the remainder a leaf factor. We consider fully general star factorings in our definitions and stubborn set methods. In what follows, assume a star factoring $\mathcal{F}$.

Actions affecting the center are **center actions**, actions affecting a leaf are **leaf actions**. We denote the set of all center actions by $\mathcal{A}^C$, and the set of all leaf actions affecting a leaf F^L by $\mathcal{A}^L[F^L]$. Where useful for readability, we will indicate center actions by the notation convention a^C, and leaf actions by the notation convention a^L. Observe that, in fork and inverted-fork factorings, the two kinds of actions are disjoint, i.e., $\mathcal{A}^C \cap \mathcal{A}^L[F^L] = \emptyset$. For general star factorings, this is not so as a center action may also affect one or several leaf factors.

A **center path** is a sequence π^C of center actions applicable to s_0 in the projection onto F^C (i.e., when removing all other variables from Π); the outcome of this application, which is a **center state**, i.e., a value assignment to F^C, is called the path's **end state**. Similarly, a **leaf path** is a sequence of leaf actions applicable to s_0 in the projection onto F^L, and its end state, a **leaf state**, is defined accordingly.

A leaf path π^L for leaf F^L **complies** with π^C – also: π^L is π^C-**compliant** – if its $\mathcal{A}^L[F^L] \cap \mathcal{A}^C$ subsequence is identical to that of π^C, and its $\mathcal{A}^L[F^L] \setminus \mathcal{A}^C$ subsequence can be scheduled alongside π^C so that the combined action sequence is applicable to s_0 in the projection onto $F^C \cup F^L$. Intuitively, the compliant leaf paths are the possible leaf moves given π^C.

For illustration, in our Vanilla example the leaf path $\pi^L = \langle$*load*(p_1, t, A),*unload*$(p_1, t, B)\rangle$ complies with the center path $\pi^C = \langle$*drive*$(A, B)\rangle$, and the leaf path $\pi^{L'} = \langle\rangle$ complies with that

center path as well. In the NoEmpty variant, setting $\pi^C = \langle \mathit{drive}(A, B, p_1) \rangle$, π^L is π^C-compliant but $\pi^{L\prime}$ is not: the precondition $p_1 = T$ of $\mathit{drive}(A, B, p_1)$ is provided on π^L, but is not provided on $\pi^{L\prime}$.

Observe that, given π^C, the possible moves of each leaf are independent across leaves: *for each leaf factor we can choose an arbitary π^C-compliant leaf path, regardless of the choice made for any other leaf factor, and still obtain a combined action sequence applicable to s_0*. In our example, given a truck-move sequence, each package can move independently of the other packages so long as it complies with the truck moves. E.g., denote $\pi^L_i := \langle \mathit{load}(p_i, t, A), \mathit{unload}(p_i, t, B) \rangle$ and $\pi^{L\prime}_i = \langle\rangle$. Then, given $\pi^C = \langle \mathit{drive}(A, B) \rangle$ in the Vanilla example, each leaf factor $\{p_i\}$ can choose either of these two compliant options regardless of the choice made for any other $\{p_j\}$, and scheduling all these choices alongside π^C yields a combined action sequence applicable to s_0.

Decoupled search exploits this by searching over center paths π^C only, keeping track of the π^C-compliant paths separately for each leaf. Concretely, the search is over **decoupled states** $s^\mathcal{F}$. Such a state is uniquely identified by the center path $\pi^C[s^\mathcal{F}]$ on which it is reached. It is associated with: its center state $\mathsf{center}[s^\mathcal{F}]$, the end state of $\pi^C[s^\mathcal{F}]$; and its **pricing function** $\mathsf{prices}[s^\mathcal{F}]$. The latter maps each leaf state s^L to its **price** $\mathsf{prices}[s^\mathcal{F}](s^L)$, defined as the cost of a cheapest $\pi^C[s^\mathcal{F}]$-compliant leaf path that ends in s^L, or $\mathsf{prices}[s^\mathcal{F}](s^L) = \infty$ if no such path exists.

For illustration, $\pi^C = \langle \mathit{drive}(A, B) \rangle$ in our Vanilla example leads to $s^\mathcal{F}$ where $\mathsf{center}[s^\mathcal{F}] = \{t = B\}$; $\mathsf{prices}[s^\mathcal{F}](\{p_i = A\}) = 0$ due to the $\pi^C[s^\mathcal{F}]$-compliant leaf path $\langle\rangle$; $\mathsf{prices}[s^\mathcal{F}](\{p_i = T\}) = 1$ due to the $\pi^C[s^\mathcal{F}]$-compliant leaf path $\langle \mathit{load}(p_i, t, A) \rangle$; and $\mathsf{prices}[s^\mathcal{F}](\{p_i = B\}) = 2$ due to the $\pi^C[s^\mathcal{F}]$-compliant leaf path $\langle \mathit{load}(p_i, t, A), \mathit{unload}(p_i, t, B) \rangle$. In the NoEmpty example, $\pi^C = \langle \mathit{drive}(A, B, p_1) \rangle$ leads to the same center state and pricing function, except that $\mathsf{prices}[s^\mathcal{F}](\{p_1 = A\}) = \infty$ as $\langle\rangle$ is not $\pi^C[s^\mathcal{F}]$-compliant here for p_1 (it does not provide the precondition $p_1 = T$ of $\mathit{drive}(A, B, p_1)$, cf. above).

One can think of a decoupled state $s^\mathcal{F}$ as a set of states s, namely those s sharing the center state $\mathsf{center}[s^\mathcal{F}]$ and where each leaf has a leaf state s^L such that $\mathsf{prices}[s^\mathcal{F}](s^L) < \infty$. These are exactly the states s that can be reached from s_0 on some action sequence with the center-action subsequence $\pi^C[s^\mathcal{F}]$.

The pricing function for a leaf F^L can be maintained in time low-order polynomial in the size of F^L's state space (the state space of the projection onto F^L). We omit the details as they are not relevant to our contribution.

The word "price" is chosen intentionally, as this is not a "cost" we have already spent. Rather, the actual commitment to a π^C-compliant path is made only at plan extraction time. The price of a leaf state s^L is the cost we *will* have to spend if, at plan extraction time, we decide to use s^L.

Decoupled search starts in the **initial decoupled state** $s^\mathcal{F}_0$, whose center path is empty $\pi^C[s^\mathcal{F}_0] = \langle\rangle$, whose center state $\mathsf{center}[s^\mathcal{F}_0]$ is $s_0[F^C]$, and whose pricing function $\mathsf{prices}[s^\mathcal{F}_0]$ results from those leaf paths applicable given $s_0[F^C]$. In our examples (both Vanilla and NoEmpty), the price is 0 for each $p_i = A$; is 1 for each $p_i = T$; and is ∞ for each $p_i = B$ as, without a truck/center move, the packages cannot get to B. In other words, the leaf states $\{p_i = B\}$ are not *reached* in $s^\mathcal{F}_0$.

More generally, given a partial variable assignment p, we say that p is **reached** in a decoupled state $s^\mathcal{F}$ if $\mathsf{center}[s^\mathcal{F}] \models p[F^C]$, and for each leaf F^L there exists a leaf state s^L such that $s^L \models p[F^L]$ and $\mathsf{prices}[s^\mathcal{F}](s^L) < \infty$. (Note the special case where p is not defined on F^C, respectively F^L: then $p[F^C]$, respectively $p[F^L]$, is the empty assignment $\emptyset$, which is satisfied by any other partial assignment.)

We say that an action a is **reached** in $s^\mathcal{F}$ if $pre(a)$ is reached in $s^\mathcal{F}$. Expanding $s^\mathcal{F}$ in decoupled search means to apply all reached center actions a^C (recall that we do not branch over leaf actions). Such an application results in $t^\mathcal{F}$ where $\pi^C[t^\mathcal{F}] := \pi^C[s^\mathcal{F}] \circ \langle a^C \rangle$, and $\mathsf{center}[t^\mathcal{F}]$ as well as $\mathsf{prices}[t^\mathcal{F}]$ arise from $\pi^C[t^\mathcal{F}]$ as defined above. In our Vanilla example, applying $\mathit{drive}(A, B)$ to $s_0^\mathcal{F}$, in the resulting $t^\mathcal{F}$ the center state is $\mathsf{center}[t^\mathcal{F}] = \{t = B\}$, and the pricing function $\mathsf{prices}[t^\mathcal{F}]$ has value 0 for each $p_i = A$, 1 for each $p_i = T$, and 2 for each $p_i = B$.

The **decoupled state space** is the space of all decoupled states reachable from $s_0^\mathcal{F}$, along with the corresponding decoupled-state transitions. A **goal decoupled state** $s_\star^\mathcal{F}$ is one where $s_\star$ is reached, like $\{p_1 = B, \ldots, p_n = B\}$ in the decoupled state $t^\mathcal{F}$ just described. From such an $s_\star^\mathcal{F}$, a plan π for the input task Π can be extracted by augmenting the center path $\pi^C[s_\star^\mathcal{F}]$ with, for every leaf F^L, a $\pi^C[s_\star^\mathcal{F}]$-compliant leaf path ending in a goal leaf state $s_\star^L$, i.e., in a leaf state satisfying $s_\star[F^L]$. In other words, we now select goal leaf paths, and commit to them. In $s_\star^\mathcal{F} = t^\mathcal{F}$ as just described, for each p_i we select the compliant leaf path $\langle \mathit{load}(p_i, t, A), \mathit{unload}(p_i, t, B) \rangle$, leading to the plan π that loads all packages, drives to B, and unloads all packages.

Selecting, for the plan π, the *cheapest* compliant paths ending in the goal leaf states $s_\star^L$, by construction we have $\mathit{cost}(\pi) = \mathit{cost}(\pi^C[s_\star^\mathcal{F}]) + \sum_{F^L \in \mathcal{F}^L} \mathsf{prices}[s_\star^\mathcal{F}](s_\star^L)$. If we select, for each F^L, an $s_\star^L$ with *minimal* $\mathsf{prices}[s_\star^\mathcal{F}](s_\star^L)$ among all goal leaf states for F^L, then π is guaranteed to be optimal among the plans for Π whose center action subsequence is $\pi^C[s_\star^\mathcal{F}]$. We denote by $\mathsf{goalprice}[s_\star^\mathcal{F}](F^L)$ the (minimal) **goal-price** of a leaf factor F^L in a goal decoupled state $s_\star^\mathcal{F}$, formally $\mathsf{goalprice}[s_\star^\mathcal{F}](F^L) = \min_{s^L : s^L \models s_\star[F^L]} \mathsf{prices}[s_\star^\mathcal{F}](s^L)$.

Observe that extensions of the center action subsequence $\pi^C[s_\star^\mathcal{F}]$ may lead to cheaper plans overall, because, with additional center actions, cheaper leaf paths may become available.[3] To guarantee optimality, decoupled search must hence continue the search on goal decoupled states. Prior work has shown that standard search algorithms can be easily adapted, by encoding the cost of the compliant leaf goal paths, in a given goal decoupled state $s_\star^\mathcal{F}$, as an additional outgoing state transition from $s_\star^\mathcal{F}$, which must be taken to end the search. For our purposes here, the main implication is that trying to reach the goal (non-goal decoupled state) is different from trying to decrease the prices of compliant goal leaf paths (goal decoupled state). Our stubborn set constructions will distinguish these two cases.

Stubborn set construction requires to analyse goal achieval starting from a given state. To do so for decoupled states $s^\mathcal{F}$, we will need the following concept of **decoupled plans**, not introduced in this form previously. Given $s^\mathcal{F}$, a decoupled plan is a transition path $\pi^\mathcal{F}$ in the decoupled state space, leading from $s^\mathcal{F}$ to a goal decoupled state $s_\star^\mathcal{F}$. The center-action sequence underlying $\pi^\mathcal{F}$ is denoted by $\pi^C[\pi^\mathcal{F}]$. To link decoupled plans $\pi^\mathcal{F}$ to actual plans for the input task Π, we will consider the **global plans**, i.e. the plans π for Π, corresponding to $\pi^\mathcal{F}$: a global plan given $\pi^\mathcal{F}$ is any π extracted from $s_\star^\mathcal{F}$ using cheapest compliant paths ending in minimum-price goal leaf states. Observe that all global plans π given $\pi^\mathcal{F}$ have the same cost, namely the optimal cost of any plan for Π using the center-action subsequence $\pi^C[s_\star^\mathcal{F}] = \pi^C[s^\mathcal{F}] \circ \pi^C[\pi^\mathcal{F}]$. We say that $\pi^\mathcal{F}$ is optimal if the cost of the global plans given $\pi^\mathcal{F}$ is minimal among all decoupled plans for $s^\mathcal{F}$. We further say that $\pi^\mathcal{F}$ is strongly optimal if it is optimal and $\pi^C[\pi^\mathcal{F}]$ contains the minimal number of 0-cost actions among all optimal decoupled plans for $s^\mathcal{F}$.

3. For example, say that in $s^\mathcal{F}$ the leaf goal *have-car* has price 20000 via the applicable leaf action *buy-car*. But if we apply a center action *get-manager-job*, then the leaf action *get-company-car* becomes applicable, reducing the leaf goal price to 0.

3.2 Strong Stubborn Sets

Strong stubborn set (SSS) pruning is a form of partial-order reduction in state space search, exploiting action commutativity to safely prune a subset of the applicable actions in a state. As previously outlined, the method analyses, given a state s, how actions enable each other to achieve some open part of the goal when starting from s, and how these actions may interfere with other actions. We briefly spell this out in what follows, in a form connecting directly to our later extensions for decoupled search.

We use a *syntactic* characterization of stubborn sets, based directly on the syntactic specifications of actions in the input task, as opposed to a *semantic* characterization relying on properties of action executions in particular states. The syntactic characterization is coarser, but is simpler and is what we use in practice.

Let s be a solvable non-goal state, $s \not\models s_\star$, and let $\mathcal{A}_s \subseteq \mathcal{A}$ be a set of actions. We say that $\mathcal{A}_s$ is **safe** for s if there exists a strongly optimal plan $\pi = \langle a_1, \ldots, a_n \rangle$ for s such that $a_1 \in \mathcal{A}_s$. We need π to be *strongly* optimal, because otherwise the search can become incomplete in the presence of 0-cost actions.[4] Note that we need to take care only of solvable states here as, on unsolvable states, any pruning method (any subset of applicable actions) is optimality-preserving.

Strong stubborn sets are a means to derive safe sets $\mathcal{A}_s$. Towards this, two basic notions are employed:

- Given a partial assignment p where $s \not\models p$, an action set $A \subseteq \mathcal{A}$ is a **necessary enabling set** for p in s if there exists $v \in \mathit{vars}(p)$ such that $s[v] \neq p[v]$ and $A = \{a' \in \mathcal{A} \mid \mathit{eff}(a')[v] = p[v]\}$.
- Given $a, a' \in \mathcal{A}$, we say that a and a' *interfere* if $\mathit{eff}(a) \not\parallel \mathit{pre}(a')$ or $\mathit{eff}(a') \not\parallel \mathit{pre}(a)$ or $\mathit{eff}(a) \not\parallel \mathit{eff}(a')$.

In other words, a necessary enabling set for p is the set of achievers of one part of p that is still open; a and a' interfere if one disables the other's precondition, or they have conflicting effects.

These two notions can be combined in a simple recursive way to derive a safe action set for any non-goal state s. Namely, an action set $\mathcal{A}_s$ is a **strong stubborn set (SSS)** for s if the following conditions hold:

(i) $\mathcal{A}_s$ contains a necessary enabling set for $s_\star$ in s.

(ii) For all actions $a \in \mathcal{A}_s$ not applicable to s, $\mathcal{A}_s$ contains a necessary enabling set for $\mathit{pre}(a)$ in s.

(iii) For all actions $a \in \mathcal{A}_s$ applicable to s, $\mathcal{A}_s$ contains all actions a' interfering with a and where $\mathit{pre}(a) \parallel \mathit{pre}(a')$.

The proof that such $\mathcal{A}_s$ is safe was originally given in planning by Alkhazraji et al. (2012), and was later enhanced by Wehrle and Helmert (2014) allowing (amongst other things) reachability information, captured above in the requirement $\mathit{pre}(a) \parallel \mathit{pre}(a')$ in (iii). We include the proof here as our later proofs for decoupled strong stubborn sets will be extensions thereof.

Consider some plan $\pi = \langle a_1, \ldots, a_n \rangle$ for s. Then the following properties hold:

4. This happens in the following example: Let s, s' be two solvable non-goal states such that $s[\![a_1]\!] = s'$, and $s'[\![a_2]\!] = s$, where $\mathit{cost}(a_1) = \mathit{cost}(a_2) = 0$. Then a_1 starts an optimal plan in s and a_2 in s', but the action sets $\mathcal{A}_s = \{a_1\}$ and $\mathcal{A}_{s'} = \{a_2\}$ prune all solutions for s and s'.

(a) There exists an action shared between $\mathcal{A}_s$ and π, i.e., $\mathcal{A}_s \cap \{a_1, \dots, a_n\} \neq \emptyset$.

This is because by (i) $\mathcal{A}_s$ contains a necessary enabling set for $s_\star$ in s.

Given (a), let a_k be the shared action with smallest index, i.e., say that $a_k \in \mathcal{A}_s$ and $\{a_1, \dots, a_{k-1}\} \cap \mathcal{A}_s = \emptyset$.

(b) a_k is applicable to s.

This is because, otherwise, by (ii) $\mathcal{A}_s$ contains a necessary enabling set A for $pre(a_k)$ in s, and one $a \in A$ must precede a_k on π, in contradiction to a_k being the first shared action.

(c) a_k does not interfere with any of the preceding actions a_i, $1 \leq i \leq k-1$, where $pre(a) \parallel pre(a_i)$.

This is because, if it did, then by (iii) we would have $a_i \in \mathcal{A}_s$, again in contradiction to a_k being the first shared action.

(d) a_k can be moved to the front of π, i.e., $\pi' := \langle a_k, a_1, \dots, a_{k-1}, a_{k+1}, \dots, a_n \rangle$ is a plan for s.

To see this, consider that, by (b), a_k is applicable in s. As a_1 is applicable in s as well, we must have $pre(a_k) \parallel pre(a_1)$, so by (c) a_1 does not interfere with a_k, and hence a_k is still applicable in $s[\![a_1]\!]$. But then, the same argument applies to a_2 and $s[\![a_1]\!]$, so iterating the argument we obtain that, for $1 \leq i \leq k-1$, $pre(a_k) \parallel pre(a_i)$ and a_i does not interfere with a_k.

The claim follows directly from (d).

4. Strong Stubborn Sets for Star-Topology Decoupled Search

We now show how to apply the idea of strong stubborn set construction to decoupled search, for arbitrary star topologies. Section 4.1 introduces basic concepts used in all our constructions, lifting, amongst others, the notions of safety and necessary enabling sets to decoupled search. Section 4.2 then introduces decoupled strong stubborn sets for non-goal decoupled states $s^{\mathcal{F}}$, where $s_\star$ has yet to be reached; and Section 4.3 shows how to handle goal decoupled states $s^{\mathcal{F}}_\star$, where $s_\star$ is already reached yet may be reached with cheaper compliant leaf paths below $s^{\mathcal{F}}_\star$. In each of these two sections, we prove the safety of our constructions.

Throughout, we assume an FDR task $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ and a star factoring $\mathcal{F}$ with center F^C and leaves $F^L \in \mathcal{F}^L$.

4.1 Basic Concepts

In the standard setting, i.e., in forward state space search, safety of an action set $\mathcal{A}_s$ for a non-goal state s means that there exists a strongly optimal plan for s starting with an action from $\mathcal{A}_s$. For decoupled search, i.e., decoupled states $s^{\mathcal{F}}$, this definition changes in two ways. First, instead of plans for s we need to talk about decoupled plans for $s^{\mathcal{F}}$. Second, as decoupled search has to continue below $s^{\mathcal{F}}$ even if $s^{\mathcal{F}}$ is a goal decoupled state, instead of "non-goal" we need to say that the empty decoupled plan is not optimal for $s^{\mathcal{F}}$. We hence define:

Definition 1 (Safety). *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}$ be a solvable decoupled state for which $\langle\rangle$ is not an optimal decoupled plan, and let $\mathcal{A}^{\mathcal{F}}_s \subseteq \mathcal{A}$ be a set of center actions.*

*We say that $\mathcal{A}_s^{\mathcal{F}}$ is **safe** for $s^{\mathcal{F}}$ if there exists a strongly optimal decoupled plan $\pi^{\mathcal{F}}$ for $s^{\mathcal{F}}$ such that $\pi^C[\pi^{\mathcal{F}}] = \langle a_1^C, \dots, a_n^C \rangle$ where $a_1^C \in \mathcal{A}_s^{\mathcal{F}}$.*

Clearly, a safe $\mathcal{A}_s^{\mathcal{F}}$ for $s^{\mathcal{F}}$ plays the same role in search as a safe $\mathcal{A}_s$ for s: pruning outgoing transitions from $s^{\mathcal{F}}$ induced by actions outside $\mathcal{A}_s^{\mathcal{F}}$ preserves optimality and completeness of the search. Note that, if $\langle\rangle$ is an optimal decoupled plan for $s^{\mathcal{F}}$, then any pruning method preserves optimality and completeness on $s^{\mathcal{F}}$, so such $s^{\mathcal{F}}$ can be disconsidered in the analysis of safety (though not in the generation of stubborn sets, as we can't recognize such states up front).

To identify safe $\mathcal{A}_s^{\mathcal{F}}$, we will rely on commutativity, i.e., on plan permutations. Whereas previously this simply referred to plans for s, the notion of permutations now becomes significantly more complicated, as we have to consider both, decoupled plans and their associated global plans.

Recall from Section 3.1 that, given a decoupled state $s^{\mathcal{F}}$, a decoupled plan $\pi^{\mathcal{F}}$ for $s^{\mathcal{F}}$ merely is a path in a reformulated search space (the decoupled state space). The meaning of $\pi^{\mathcal{F}}$ for the actual input task is defined in terms of the global plans π given $\pi^{\mathcal{F}}$. These consist of the center action sequence $\pi^C[s^{\mathcal{F}}] \circ \pi^C[\pi^{\mathcal{F}}]$, augmented with a cheapest compliant goal leaf path π^L for each leaf factor F^L. We define permutations over both, $\pi^{\mathcal{F}}$ and π simultaneously:

Definition 2 (Plan Permutation)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}$ be a solvable decoupled state, and let $\pi^{\mathcal{F}}$ and $\pi^{\mathcal{F}'}$ be decoupled plans for $s^{\mathcal{F}}$.*

*We say that $\pi^{\mathcal{F}}$ and $\pi^{\mathcal{F}'}$ are **permutations** if (i) $\pi^C[\pi^{\mathcal{F}}]$ is a permutation of $\pi^C[\pi^{\mathcal{F}'}]$, and (ii) there exist global plans π and π' for $\pi^{\mathcal{F}}$, respectively $\pi^{\mathcal{F}'}$, such that π is a permutation of π'.*

Here (ii) essentially says that, on top of the center paths being commutative, the corresponding cheapest compliant goal leaf paths need to be commutative as well. For safety, (i) would be enough in principle, i.e., it would be enough to identify a center action starting a permutation of a strongly optimal decoupled plan. Our construction of strong stubborn sets however, i.e., our sufficient criterion for permutability, relies on both (i) and (ii).

A second major implication of the interplay between decoupled plans and their global plans is that we have to think carefully about what is the "future" for a decoupled state $s^{\mathcal{F}}$. Stubborn set construction reasons about the future, which in the standard setting simply are the possible plans π for the state s in question. But what is the future for $s^{\mathcal{F}}$? A priori, of course the possible decoupled plans $\pi^{\mathcal{F}}$ for $s^{\mathcal{F}}$. Yet, $\pi^{\mathcal{F}}$ merely is a path in a reformulated search space. An associated global plan π is not an action sequence "starting in $s^{\mathcal{F}}$": π starts in the initial state of the input planning task. In particular, the goal leaf paths π^L in π schedule actions along both, the center action subsequence $\pi^C[s^{\mathcal{F}}]$ up to $s^{\mathcal{F}}$, and the center action subsequence $\pi^C[\pi^{\mathcal{F}}]$ behind $s^{\mathcal{F}}$.

Our natural solution to this issue is to consider the "past" as everything scheduled along $\pi^C[s^{\mathcal{F}}]$, and the "future" as everything scheduled along $\pi^C[\pi^{\mathcal{F}}]$. Precisely, say that π is a global plan given $\pi^{\mathcal{F}}$, and $\pi = \langle a_1, \dots, a_n \rangle$. Consider the center action subsequence $\pi^C[s^{\mathcal{F}}] \circ \pi^C[\pi^{\mathcal{F}}]$ in π, and consider, within that subsequence, the starting action of $\pi^C[\pi^{\mathcal{F}}]$. Denote by t the index of that action occurence in π, i.e., a_t is the start of $\pi^C[\pi^{\mathcal{F}}]$ within π. We capture the "past" as $\pi^{past} := \langle a_1, \dots, a_{t-1} \rangle$, and the "future" as $\pi^{future} := \langle a_t, \dots, a_n \rangle$. In other words, we consider the first center action behind $s^{\mathcal{F}}$ – the center action decoupled search applies in $s^{\mathcal{F}}$ to find $\pi^{\mathcal{F}}$ – and we use that action as the pivot separating the past from the future. We will be using the notations a_t, π^{past}, and π^{future} throughout.

While this definition of the future is reasonably simple, it leaves us with another subtlety, namely that "the same" π may be scheduled in different ways, leaving us with undesirable ambiguity regarding the difference between past and future. To see this, consider any goal leaf path π^L. Observe that

the part of π^L contained within π^{past} – the leaf path's prefix scheduled in the past – must comply with $\pi^C[s^\mathcal{F}]$. Yet nothing forces this prefix to be maximal: we are free to schedule parts of π^L in the future, even if they comply with $\pi^C[s^\mathcal{F}]$ so could be scheduled in the past. For illustration, say the center (e.g. a truck) has already done up to $s^\mathcal{F}$ what it needs to do in order to support a leaf (e.g. providing transport for a package). But now the leaf needs to do additional tasks independently from the center (e.g. unpacking, assembling package content, ...). Then π *can* schedule these additional tasks in π^{past}, but may just as well schedule them in π^{future}.

In other words, π^{future} may contain actions that have nothing to do with "the remaining task starting from $s^\mathcal{F}$". It will be of service to our analyses to avoid this kind of behavior. We will restrict focus to global plans π that are *past-maximal*, where, intuitively, the leaves do nothing in π^{future} that they could do in π^{past}.

The precise form of past-maximality we will need differs depending on the context: we introduce one notion for star topologies, and another different one for our analysis of fork topologies below. In our notion for star topologies, past-maximality means that no leaf action scheduled in the future can be moved in front of the pivot action a_t:

Definition 3 (Past-Maximality). *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}$ be a solvable decoupled state, let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}$, and let π be a global plan given $\pi^\mathcal{F}$.*

*Define a_t, π^{past}, and π^{future} as above. We say that π is **past-maximal** if, for every leaf factor F^L and every $k > t$ where $a_k \in \mathcal{A}^L[F^L] \setminus \mathcal{A}^C$, if we change π by moving a_k to any position in front of a_t, then π is not a plan anymore.*

It is easy to see that a focus on past-maximal global plans is not restricting, in the sense that every global plan can be rescheduled to be past-maximal:

Lemma 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}$ be a decoupled state, let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}$, and let π be a global plan given $\pi^\mathcal{F}$.*

Then there exists a permutation of π that is a global plan given $\pi^\mathcal{F}$, and that is past-maximal.

Proof. Obtain such a permutation π' as follows. Start with $\pi' := \pi$. If π' is past-maximizing, stop. Else, select a counter-example F^L and k, move a_k in front of a_t in π', and iterate. This algorithm terminates as, after each step, there is one action less behind a_t. The outcome π' satisfies the claim as we have not changed the center-action subsequence, and the permuted leaf paths are still compliant with that sequence. □

We are now ready to introduce the ingredients underlying our stubborn set constructions. As before, we will need to reason about how actions *interfere* with each other, and how they *enable* each other. The former remains exactly the same as before. The latter, however, needs to be adapted substantially given the more complex structure of the search space. We need to distinguish between enabling (a) conditions p not reached in a decoupled state, vs. (b) conditions p reached in a decoupled state. For (b), our constructions will depend on the specific context (stars vs. forks/inverted forks, goal vs. non-goal decoupled states), so we will introduce these in the respective context. For (a), the same construction works in all contexts. Namely, we employ the following extended definition of necessary enabling sets:

Definition 4 (Decoupled Necessary Enabling Set). *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}$ be a decoupled state, and let p be a partial variable assignment not reached in $s^\mathcal{F}$.*

*An action set A is a **decoupled necessary enabling set** for p in $s^{\mathcal{F}}$ if one of the following conditions holds:*

(i) There exists $v \in vars(p) \setminus F^C$ such that $p[v]$ is not reached in $s^{\mathcal{F}}$, and $A = \{a \in \mathcal{A} \mid \mathit{eff}(a)[v] = p[v]\}$.

(ii) Condition (i) does not apply, and there exists F^L such that $p[F^L]$ is not reached in $s^{\mathcal{F}}$, and $A = \bigcup_{v \in vars(p) \cap F^L} \{a \in \mathcal{A} \mid \mathit{eff}(a)[v] = p[v]\}$.

(iii) Neither (i) nor (ii) apply, and there exists $v \in vars(p) \cap F^C$ such that $\mathsf{center}[s^{\mathcal{F}}][v] \neq p[v]$, and $A = \{a \in \mathcal{A} \mid \mathit{eff}(a)[v] = p[v]\}$.

In other words, a decoupled necessary enabling set first checks whether (i) some leaf-variable value is not reached in $s^{\mathcal{F}}$, then checks whether (ii) some leaf-factor state is not reached in $s^{\mathcal{F}}$, then checks whether (iii) some center-variable value is not reached in $s^{\mathcal{F}}$. In each case, the set of all achieving actions is selected.

As p is not reached in $s^{\mathcal{F}}$, one of (i) – (iii) must necessarily be true. Observe that this is not so for just (i) and (iii) alone, as every single leaf-variable value in p may already be reached in some leaf state of F^L, yet that may not be so for their combination. In other words, necessary enabling sets become more complicated, relative to standard search, due to the difference between leaf states being true in a unique state s, vs. leaf states being true in one out of the set of states represented by a decoupled state $s^{\mathcal{F}}$.

We remark that the ordering of conditions (i) – (iii) in Definition 4 is arbitrary in the sense that any ordering is possible in principle. The stated ordering encapsulates a preference to regress over open leaf conditions first, the aim being to extract open center conditions (which the decoupled strong stubborn set will be chosen to support) "as close as possible" to the current decoupled state $s^{\mathcal{F}}$. The intuition is to first move the leaves towards their goals – by enabling the required center preconditions – and to care for the center goals only once the leaf goals are achieved.

4.2 Non-Goal Decoupled States

Towards our constructions for non-goal decoupled states $s^{\mathcal{F}}$, we next specify how to enable a condition p *reached* in $s^{\mathcal{F}}$.

To understand why this is needed, consider the role of reached leaf states s^L in $s^{\mathcal{F}}$, where for each F^L one such s^L will be used, and that commitment will be made at plan extraction time. This implies that, just because some condition p on F^L is reached in $s^{\mathcal{F}}$, it does not have to be true at the corresponding point in the global plan π we will finally commit to. Namely, defining "the corresponding point" in π as the state $s_0[\![\pi^{past}]\!]$ before application of the pivot action a_t, p will have to be false at this point if a_t has a precondition contradicting p. Nevertheless, p may have to be true at some later point along π, to enable some other action or part of the goal.

For illustration, consider the decoupled state $s^{\mathcal{F}} := s_0^{\mathcal{F}}$ in our NoEmpty example, consider $F^L := \{p_1\}$, and consider $p := \{p_1 = A\}$. Condition p is reached in $s^{\mathcal{F}}$. Yet, say our decoupled plan $\pi^{\mathcal{F}}$ decides to use $a_t := \mathit{drive}(A, B, p_1)$ in $s^{\mathcal{F}}$. Then p must be false prior to the application of a_t in the corresponding global plan π, due to the incompatible precondition $\{p_1 = T\}$. If p is required later on, e.g. if $s_\star[p_1] = A$, then p will need to be enabled in the future, behind the application of a_t in π. We need to reason about how to do that.

So, how do we find a set of actions that will necessarily be used to enable a reached F^L condition p in the future? A simple possibility would be to follow Definition 4 (ii) and just select all actions supporting any variable value in p. Done recursively though, this is likely to collect a very large set of reached actions that affect F^L. We hence design a definition more tailored to the specific situation: a reached F^L condition p, false in some reached F^L state s^L (namely $s_0[\![\pi^{past}]\!][F^L]$), yet true in some F^L state r^L (namely one r^L visited along π^{future}), where r^L is achieved on a path $\pi^L_{s\to r}$ starting from s^L (namely the respective leaf path segment within π^{future}). It suffices to ensure that we collect at least one action along every such path $\pi^L_{s\to r}$:

Definition 5 (Reached-Enabling Set)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}$ be a decoupled state, let F^L be a leaf factor, and let p be a partial assignment to F^L reached in $s^\mathcal{F}$.*

*We say that an F^L state s^L is a **reached-enabling state** for p in $s^\mathcal{F}$ if*

(i) s^L is reached in $s^\mathcal{F}$, and $s^L \not\models p$; and

(ii) there is an F^L path $\pi^L_{s\to r}$ from s^L to an F^L state r^L where $r^L \models p$.

*An action set A is a **reached-enabling set** for p in $s^\mathcal{F}$ if every path $\pi^L_{s\to r}$ as in (ii) contains at least one action from A.*

For illustration, in our above example where $p = \{p_1 = A\}$, the only reached s^L where $s^L \not\models p$ is $s^L = \{p_1 = T\}$. That s^L has two outgoing transitions, labeled by *unload*(p_1, t, A) respectively *unload*(p_1, t, B). Intuitively, only *unload*(p_1, t, A) makes sense as an enabler for p. And indeed, that action is a reached-enabling set on its own, because every path $\pi^L_{s\to r}$ achieving p must use it eventually.

It is easy to see that our construction is correct, in the sense that a reached p not true at the end of π^{past}, yet true somewhere on π^{future}, must be enabled on π^{future} with an action from a reached-enabling set:

Lemma 2. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}$ be a solvable decoupled state, let F^L be a leaf factor, and let p be a partial assignment to F^L reached in $s^\mathcal{F}$. Let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}$, and let $\pi = \langle a_1, \ldots, a_n \rangle$ be a global plan given $\pi^\mathcal{F}$. Let A be a reached-enabling set for p in $s^\mathcal{F}$.*

Define a_t, π^{past}, and π^{future} as before. Denote the states traversed by π as $s_0, \ldots, s_n$. If $s_{t-1} \not\models p$ but there exists $k > t$ such that $s_{k-1} \models p$, then $\{a_t, \ldots, a_{k-1}\} \cap A \neq \emptyset$.

Proof. Denote by $\pi^L = \langle a^L_1, \ldots, a^L_m \rangle$ the F^L leaf path in π preceding a_k, i.e., the F^L leaf path induced by $\langle a_1, \ldots, a_{k-1} \rangle$. Say that π^L traverses the F^L states $s^L_0, \ldots, s^L_m$. Denote by l the index of the F^L state at the end of π^{past}. Denote by $\pi^L_{l\to m} := \langle a^L_{l+1}, \ldots, a^L_m \rangle$ the segment of π^L within π^{future}.

By construction, s^L_l is reached in $s^\mathcal{F}$. By prerequisite, $s^L_l \not\models p$. Furthermore, $s^L_m \models p$ because, by construction, a^L_m is the last action preceding a_k that affects F^L, so s^L_m agrees with s_{k-1} on F^L, and by prerequisite $s_{k-1} \models p$. Finally, the segment $\pi^L_{l\to m}$ of π^L is an F^L path from s^L_l to s^L_m. So s^L_l is a reached-enabling state for p in $s^\mathcal{F}$. By Definition 5, A contains at least one action a^L_i from $\pi^L_{l\to m}$, which shows the claim. □

Observe that a reached-enabling set A is essentially a cut between the node sets $\{s^L\}$ and $\{r^L\}$ in F^L's state space. One could, hence, consider to compute reached-enabling sets via minimum

cuts. Observe though that this would be optimizing the wrong objective – we want to minimize, not the number of cut-set transitions, but the number of actions $|A|$ these are labeled with. It is easy to see that optimizing that objective is **NP**-complete.[5] More importantly, reached-enabling sets will need to be computed very frequently – potentially many times on every search state – so their computation must be extremely cheap.

An alternative could be a greedy procedure considering only the actions a labeling an outgoing transition $s^L \xrightarrow{a} t^L$ of a reached-enabling state s^L, iteratively collecting and removing one such a until the node sets $\{s^L\}$ and $\{r^L\}$ are disconnected. But this also would be too expensive, requiring repeated reachability checks on the leaf state space. We therefore settle for a trivial approximation, simply collecting all actions a from the transitions $s^L \xrightarrow{a} t^L$. This makes the construction rather inclusive of course, that is, the constructed reached-enabling sets may be large (e.g. we fail to discard *unload*(p_1, t, B) in the example above). We will show below how more targeted (yet sufficiently cheap) constructions can be designed for goal decoupled states and for fork structures, where enabling p is only relevant if its price decreases strictly.

We are now ready to put the pieces together, and define strong stubborn sets for non-goal decoupled states:

Definition 6 (DSSS: Stars, Non-Goal)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}$ be a non-goal decoupled state.*

*An action set $\mathcal{A}_s^\mathcal{F}$ is a **decoupled strong stubborn set (DSSS)** for $s^\mathcal{F}$ if all of the following conditions hold:*

(i) $\mathcal{A}_s^\mathcal{F}$ contains a decoupled necessary enabling set for $s_\star$ in $s^\mathcal{F}$.

(ii) For all actions $a \in \mathcal{A}_s^\mathcal{F}$ not reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains a decoupled necessary enabling set for $pre(a)$ in $s^\mathcal{F}$.

(iii) For all actions $a \in \mathcal{A}_s^\mathcal{F}$ reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains all actions a' interfering with a and where $pre(a) \parallel pre(a')$.

(iv) For all actions $a \in \mathcal{A}_s^\mathcal{F}$ reached in $s^\mathcal{F}$, and for all F^L where $pre(a)[F^L] \neq \emptyset$, $\mathcal{A}_s^\mathcal{F}$ contains a reached-enabling set for $pre(a)[F^L]$ in $s^\mathcal{F}$.

Items (i) – (iii) of this definition are in obvious correspondence to that for strong stubborn sets in standard search (cf. Section 3.2), replacing necessary enabling sets with decoupled necessary enabling sets in (i) and (ii), and replacing applicability with being reached in (ii) and (iii). The new item (iv) is needed to cover enablers for leaf preconditions already reached in $s^\mathcal{F}$, as discussed above.

Clearly, the construction of DSSS as per Definition 6 is operational. Viewing the definition as a recursive fixed-point algorithm, all the required constructs – decoupled necessary enabling sets, interfering actions, reached-enabling sets – can be computed in time low-order polynomial in the size of the input task and its leaf state spaces, i.e., in the same size parameters as the computation of the decoupled states themselves.

It remains to prove that Definition 6 is actually correct, i.e., that a DSSS $\mathcal{A}_s^\mathcal{F}$ for $s^\mathcal{F}$ is safe for $s^\mathcal{F}$. We do so via extending the proof for strong stubborn sets as given in Section 3.2. In particular, we

5. The proof uses a reduction from Hitting Set, where each label subset $\{l_1, \ldots, l_n\}$ is represented through a separate path $\pi^L_{s \to r}$ carrying these labels.

show that for every decoupled plan $\pi^{\mathcal{F}}$ in $s^{\mathcal{F}}$, $\mathcal{A}_s^{\mathcal{F}}$ contains a center action starting a permutation of $\pi^{\mathcal{F}}$ (which by Definition 2 is also a decoupled plan). Note that we actually prove the stronger result that the two decoupled plans induce globals plans leading to the *same* goal state, which implies the claim. We include the full proof here, as the underlying analysis is key to understanding why our techniques work.

The proof consists of six successive observations (a) – (e), and a final concluding argument. Observations (a) – (c) correspond to those for the standard setting, namely that (a) there exists a shared action, (b) the first shared action a_k is applicable, (c) a_k does not interfere with the preceding actions. The only major difference here is that, in (b), we can only conclude that a_k is reached in the decoupled state $s^{\mathcal{F}}$, as opposed to being applicable in a state s. Observation (e), moving a_k up front, mirrors observation (d) for the standard setting. Observations (c) and (f), as well as the concluding argument, are new. They are required to: (c) deal with the difference between being reached vs. being applicable; (f) prove that a_k is a center action so we're actually permuting the center path and hence the decoupled plan; (concluding argument) deal with the more complex structure of permutations, i.e., the interplay between decoupled plans and global plans.

Theorem 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}$ be a solvable non-goal decoupled state, and let $\pi^{\mathcal{F}}$ be a decoupled plan for $s^{\mathcal{F}}$. Let $\mathcal{A}_s^{\mathcal{F}}$ be a DSSS for $s^{\mathcal{F}}$.*

Then $\mathcal{A}_s^{\mathcal{F}}$ contains a center action starting a permutation of $\pi^{\mathcal{F}}$.

Proof. Let π be a global plan for $\pi^{\mathcal{F}}$, and assume, without loss of generality by Lemma 1, that π is past-maximal. Denote $\pi = \langle a_1, \ldots, a_n \rangle$. As above, let a_t be the starting action of $\pi^C[\pi^{\mathcal{F}}]$ in π, and denote $\pi^{past} := \langle a_1, \ldots, a_{t-1} \rangle$ and $\pi^{future} := \langle a_t, \ldots, a_n \rangle$. Then the following properties hold:

(a) There must be an action a shared between $\mathcal{A}_s^{\mathcal{F}}$ and π^{future}, i.e., $a \in \mathcal{A}_s^{\mathcal{F}} \cap \{a_t, \ldots, a_n\}$.

First, $s_\star$ is not reached in $s^{\mathcal{F}}$, as $s^{\mathcal{F}}$ is a non-goal decoupled state. By Definition 6 (i), $\mathcal{A}_s^{\mathcal{F}}$ contains a decoupled necessary enabling set A for $s_\star$ in $s^{\mathcal{F}}$. At least one $a \in A$ must achieve some sub-assignment p of $s_\star$ that is not reached in $s^{\mathcal{F}}$. Observe that this action a cannot be scheduled on π^{past}, and hence must be on π^{future}: if p pertains to the center, this is trivial; if p pertains to a leaf, this is so because otherwise p would be reached in $s^{\mathcal{F}}$.

Given (a), let a_k be the first shared action, i.e., say that $a_k \in \mathcal{A}_s^{\mathcal{F}}$ and $\{a_t, \ldots, a_{k-1}\} \cap \mathcal{A}_s^{\mathcal{F}} = \emptyset$.

(b) a_k is reached in $s^{\mathcal{F}}$.

This is because, otherwise, by Definition 6 (ii) $\mathcal{A}_s^{\mathcal{F}}$ contains a decoupled necessary enabling set A for $pre(a_k)$ in $s^{\mathcal{F}}$. With the same argument as in (a), applied to $pre(a_k)$ instead of $s_\star$, this yields a shared action between $\mathcal{A}_s^{\mathcal{F}}$ and π^{future}. That action must precede a_k on π^{future}, in contradiction to a_k being the first shared action.

(c) a_k does not interfere with any of the actions a_i, $t \leq i \leq k-1$, where $pre(a_k) \parallel pre(a_i)$.

This is because, if it did, then by Definition 6 (iii) we would have $a_i \in \mathcal{A}_s^{\mathcal{F}}$, again in contradiction to a_k being the first shared action.

(d) The leaf precondition of a_k is true at the end of π^{past}, i.e., in $s_0[\![\pi^{past}]\!]$.

Assume to the contrary that, for some F^L, $p := pre(a_k)[F^L]$ is not true at the end of π^{past}, i.e., prior to the application of a_k in π. As π is a plan, p is true prior to the application of a_k however.

In particular, $k > t$. Furthermore, by (b) and Definition 6 (iv), $\mathcal{A}_s^{\mathcal{F}}$ contains a reached-enabling set A for p in $s^{\mathcal{F}}$. We can apply Lemma 2, and get that there exists an action $a_i \in A \subseteq \mathcal{A}_s^{\mathcal{F}}$ preceding a_k on π^{future}, again in contradiction.

(e) a_k can be moved to the start of π^{future}. Precisely, $\pi' := \pi^{past} \circ \langle a_k, a_t, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n \rangle$ is a plan for Π.

To see this, consider that, first, a_k is applicable after π^{past}, i.e., a_k is applicable to $s_0[\![\pi^{past}]\!]$: by (b) a_k is reached in $s^{\mathcal{F}}$, in particular its center precondition is true at that point; by (d), its leaf precondition is true as well. Second, as both a_k and a_t are applicable in $s_0[\![\pi^{past}]\!]$, we must have $pre(a_k) \parallel pre(a_t)$, so by (c) a_t does not interfere with a_k, and hence a_k is still applicable in $s[\![\pi^{past} \circ \langle a_t \rangle]\!]$. But then, the same argument applies to a_{t+1} and $s[\![\pi^{past} \circ \langle a_t \rangle]\!]$, so iterating the argument we obtain that, for $t \leq i \leq k-1$, $pre(a_k) \parallel pre(a_i)$ and a_i does not interfere with a_k. The claim follows directly from this.

(f) a_k is a center action.

Assume for contradiction that a_k does not affect the center, $a_k \in \mathcal{A}^L[F^L] \setminus \mathcal{A}^C$ for some F^L. As π is past-maximal, a_k cannot be moved in front of a_t in π without violating the plan property. However, the plan π' constructed as per (e) does exactly that, in contradiction.

We can now easily construct the desired permutation $\pi^{\mathcal{F}'}$ of $\pi^{\mathcal{F}}$. We construct the underlying center-action sequence $\pi^{C'}$ to be like $\pi^C[\pi^{\mathcal{F}}]$, but moving the action a_k, which by (f) is a center action so is part of $\pi^C[\pi^{\mathcal{F}}]$, to the front. Consider the plan π' constructed as per (e). The center-action subsequence of π' is $\pi^C[s^{\mathcal{F}}] \circ \pi^{C'}$. So π' consists of that center path, augmented with cheapest goal leaf paths compliant with that center path and ending in cheapest goal leaf states. Therefore, $\pi^{C'}$ induces a decoupled plan $\pi^{\mathcal{F}'}$ for $s^{\mathcal{F}}$, and the permutation π' of π is a global plan for $\pi^{\mathcal{F}'}$. This concludes the argument. □

Corollary 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}$ be a solvable non-goal decoupled state. Let $\mathcal{A}_s^{\mathcal{F}}$ be a DSSS in $s^{\mathcal{F}}$. Then $\mathcal{A}_s^{\mathcal{F}}$ is safe for $s^{\mathcal{F}}$.*

4.3 Goal Decoupled States

Let us now consider goal decoupled states $s_\star^{\mathcal{F}}$. The only thing that needs to change, relative to the definition of DSSS for non-goal decoupled states, is item (i): How to ensure that there is a shared action, i.e., that at least one action from π^{future} is contained in $\mathcal{A}_s^{\mathcal{F}}$?

For non-goal decoupled states, this question was easily answered in terms of a decoupled necessary enabling set, supporting some part p of $s_\star$ not reached in $s^{\mathcal{F}}$. In a goal decoupled state $s_\star$, however, such p does not exist. We instead need to capture all ways in which the goal price of some leaf may still be improved. We do so in terms of what we call *goal-price frontier* action sets:

Definition 7 (Goal-Price Frontier Set)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s_\star^{\mathcal{F}}$ be a goal decoupled state, and let F^L be a leaf factor where $s_\star[F^L] \neq \emptyset$.*

*We say that an F^L transition $s^L \xrightarrow{a} t^L$ is a **goal-price frontier transition** for F^L in $s_\star^{\mathcal{F}}$ if*

(i) s^L is reached in $s_\star^{\mathcal{F}}$ and $\mathsf{prices}[s_\star^{\mathcal{F}}](s^L) + cost(a) < \mathsf{prices}[s_\star^{\mathcal{F}}](t^L)$; and

(ii) $s^L \xrightarrow{a} t^L$ is part of an F^L path π^L from $s_0[F^L]$ to an F^L state r^L where $r^L \models s_\star[F^L]$ and $cost(\pi^L) < \mathsf{goalprice}[s_\star^{\mathcal{F}}](F^L)$.

*An action set A is a **goal-price frontier set** for F^L in $s^{\mathcal{F}}_\star$ if, for every path π^L as in (ii), the segment $\pi^L_{s\to r}$ of π^L starting with $s^L \xrightarrow{a} t^L$ contains at least one action from A.*

Example 2. *Figure 1 shows the leaf state space of the slightly modified example with the car and the manager, where $\mathcal{V} = \{job, have\text{-}car, location\}$, with $\mathcal{D}(job) = \{worker, manager\}$, $\mathcal{D}(have\text{-}car) = \{T, F\}$, and $\mathcal{D}(location) = \{A, B\}$. The actions are $\mathcal{A} = \{get\text{-}manager\text{-}job, walk, drive, get\text{-}company\text{-}car, buy\text{-}car\}$, where $get\text{-}manager\text{-}job$ changes the value of the variable job from $worker$ to $manager$ for cost 1, $walk$ and $drive$ set position from A to B at a cost of 100, respectively 1, where $drive$ has the additional precondition $\{have\text{-}car{=}T\}$. The actions $buy\text{-}car$ and $get\text{-}company\text{-}car$ set $have\text{-}car$ from F to T at a cost of 20000, respectively 0, where $get\text{-}company\text{-}car$ has the additional precondition $\{job{=}manager\}$. The initial state is $s_0 = \{job{=}worker, have\text{-}car{=}F, location{=}A\}$, the goal is $s_\star = \{location{=}B\}$.*

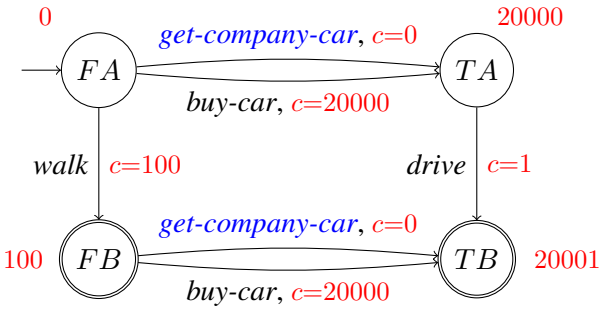

Figure 1: Illustration of the leaf state space of the task in Example 2. The actions in the goal-price frontier set are highlighted in blue.

We set the factoring $\mathcal{F}$ with a single leaf to $F^C = \{job\}$ and $F^L = \{have\text{-}car, location\}$. The pricing function of the initial decoupled state $s^{\mathcal{F}}_0$ is shown in the red numbers next to the states in Figure 1, where we abbreviate, e.g., state $\{have\text{-}car = \mathbf{F}, location = \mathbf{A}\}$ by FA. Observe that $s^{\mathcal{F}}_0$ is a goal decoupled state, where the extracted global plan is $\langle walk\rangle$ with a cost of 100. The goal-price frontier set for $s^{\mathcal{F}}_0$ is $A = \{get\text{-}company\text{-}car\}$, which captures the only way to reduce the goal price. In terms of the definition, both FA and FB qualify as s^L, where the respective t^L are TA and TB.

We use the goal-price frontier sets for all leaf factors that have a goal to ensure progress towards a cheaper *goal decoupled state (replacing the decoupled necessary enabling set for the goal). In our example, the action $get\text{-}company\text{-}car$ in the goal-price frontier set is not applicable because of the unsatisfied center precondition $\{have\text{-}car = T\}$. This leads to the center action $get\text{-}manager\text{-}job$ being added to $\mathcal{A}^{\mathcal{F}}_s$. Applying it to $s^{\mathcal{F}}_0$ results in the state $s^{\mathcal{F}} := s^{\mathcal{F}}_0[\![get\text{-}manager\text{-}job]\!]$ which has a global plan $\langle get\text{-}manager\text{-}job, get\text{-}company\text{-}car, drive\rangle$ with cost 2.*

Similarly as for reached-enabling sets above, we prove that this construction indeed captures all potential enabling actions. From this property, the proof of safety will follow immediately. As the construction of a frontier is more selective than that of reached-enabling sets – targeted at price-improving transitions – its correctness proof is more complicated though.

Lemma 3. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}_\star$ be a goal decoupled state, and let F^L be a leaf factor where $s_\star[F^L] \neq \emptyset$. Let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}_\star$, and let π be a global plan given $\pi^\mathcal{F}$. Let A be a goal-price frontier set for F^L in $s^\mathcal{F}_\star$.*

Define a_t, π^{past}, and π^{future} as before, and denote the F^L path within π by π^L. If $cost(\pi^L) < \mathsf{goalprice}[s^\mathcal{F}_\star](F^L)$, then π^{future} contains an action from A.

Proof. Denote by $s^L_0, \dots, s^L_m$ the leaf states traversed by π^L. As π^L strictly decreases the goal price for F^L, π^L cannot be fully contained in π^{past}. Denote by l the index of the first action on π^L where a^L_l is on π^{future}.

By construction, s^L_m is a goal leaf state for F^L, and $cost(\pi^L) < \mathsf{goalprice}[s^\mathcal{F}_\star](F^L)$. In particular, $cost(\pi^L) = cost(\langle a^L_1, \dots, a^L_m \rangle) < \mathsf{prices}[s^\mathcal{F}_\star](s^L_m)$, and hence there exists an index $i \geq l$ where $cost(\langle a^L_1, \dots, a^L_i \rangle) < \mathsf{prices}[s^\mathcal{F}_\star](s^L_i)$. Let i be the smallest such index. Consider the transition $s^L_{i-1} \xrightarrow{a^L_i} s^L_i$ on π^L. As $i \geq l$, this transition is part of π^{future}.

Because i is the smallest index where $cost(\langle a^L_1, \dots, a^L_i \rangle) < \mathsf{prices}[s^\mathcal{F}_\star](s^L_i)$, it follows that $\mathsf{prices}[s^\mathcal{F}_\star](s^L_{i-1}) \leq cost(\langle a^L_1, \dots, a^L_{i-1} \rangle)$. Observe that, hence, s^L_{i-1} is reached in $s^\mathcal{F}_\star$, as its price is upper-bounded by a finite value. Further, observe that, adding the cost of a^L_i on both sides of the inequality $\mathsf{prices}[s^\mathcal{F}_\star](s^L_{i-1}) \leq cost(\langle a^L_1, \dots, a^L_{i-1} \rangle)$, we get $\mathsf{prices}[s^\mathcal{F}_\star](s^L_{i-1}) + cost(a^L_i) \leq cost(\langle a^L_1, \dots, a^L_i \rangle)$. As, by construction, $cost(\langle a^L_1, \dots, a^L_i \rangle) < \mathsf{prices}[s^\mathcal{F}_\star](s^L_i)$, we obtain that $\mathsf{prices}[s^\mathcal{F}_\star](s^L_{i-1}) + cost(a^L_i) < \mathsf{prices}[s^\mathcal{F}_\star](s^L_i)$. Furthermore, $s^L_{i-1} \xrightarrow{a^L_i} s^L_i$ lies on the F^L path π^L from $s_0[F^L]$ to s^L_m where, as already observed, $s^L_m \models s_\star[F^L]$ and $cost(\pi^L) < \mathsf{goalprice}[s^\mathcal{F}_\star](F^L)$. Putting these observations together, $s^L_{i-1} \xrightarrow{a^L_i} s^L_i$ is a goal-price frontier transition for F^L in $s^\mathcal{F}_\star$.

Therefore, as A is a goal-price frontier set for F^L in $s^\mathcal{F}_\star$, we know that the segment $\langle a^L_i, \dots, a^L_m \rangle$ of π^L between s^L_{i-1} and s^L_m must contain an action $a^L_j \in A$. The claim follows as $\langle a^L_i, \dots, a^L_m \rangle$ is fully contained in π^{future}. □

How to compute a goal-price frontier set? Like for reached-enabling sets, this computation is highly time-critical, so we settle for a similar very simple solution: We collect all actions a labeling the goal-price frontier transitions $s^L \xrightarrow{a} t^L$. Note that this is more targeted still than our solution for reached-enabling sets, as the definition of goal-price frontier transitions is more restrictive. For effective implementation, we precompute, within every leaf state space and for every leaf state s^L, the minimum cost $g^*(s^L)$ to reach s^L from $s_0[F^L]$, and the minimum cost $h^*(s^L)$ to reach $s_\star[F^L]$ from s^L. Given $s^\mathcal{F}_\star$, the goal-price frontier transitions then are exactly those $s^L \xrightarrow{a} t^L$ that qualify for Definition 7 (i) which is cheap to test, and where $g^*(s^L) + cost(a) + h^*(t^L) < \mathsf{goalprice}[s^\mathcal{F}_\star](F^L)$.

We can now state the definition of DSSS for goal decoupled states very easily:

Definition 8 (DSSS: Stars, Goal)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^\mathcal{F}_\star$ be a goal decoupled state.*

*An action set $\mathcal{A}^\mathcal{F}_s$ is a **decoupled strong stubborn set (DSSS)** for $s^\mathcal{F}_\star$ if all of the following conditions hold:*

(i) For every F^L where $s_\star[F^L] \neq \emptyset$, $\mathcal{A}^\mathcal{F}_s$ contains a goal-price frontier set for F^L in $s^\mathcal{F}_\star$.

(ii) – (iv) as for non-goal decoupled states, Definition 6, replacing $s^\mathcal{F}$ with $s^\mathcal{F}_\star$.

The proof of safety also is very similar to that for non-goal decoupled states, i.e., that of Theorem 1:

Theorem 2. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}_\star$ be a goal decoupled state for which $\langle\rangle$ is not an optimal decoupled plan, and let $\pi^{\mathcal{F}}$ be a strongly optimal decoupled plan for $s^{\mathcal{F}}_\star$. Let $\mathcal{A}^{\mathcal{F}}_s$ be a DSSS for $s^{\mathcal{F}}_\star$.*

Then $\mathcal{A}^{\mathcal{F}}_s$ contains a center action starting a permutation of $\pi^{\mathcal{F}}$.

Proof. The only difference to the proof of Theorem 1 is in argument (a), showing that there is a shared action a contained in both π^{future} and $\mathcal{A}^{\mathcal{F}}_s$.

Observe that, with $\langle\rangle$ not being an optimal decoupled plan, $\pi^{\mathcal{F}}$ must strictly decrease the price of the goal for at least one leaf factor F^L. That is, there must exist F^L where $s_\star[F^L] \neq \emptyset$ and, denoting the F^L path within π by π^L, $cost(\pi^L) < \mathsf{goalprice}[s^{\mathcal{F}}_\star](F^L)$. By Definition 8 (i) $\mathcal{A}^{\mathcal{F}}_s$ contains a goal-price frontier set A for F^L in $s^{\mathcal{F}}_\star$. We can apply Lemma 3, and get that π^{future} contains an action from A, proving the claim. □

Corollary 2. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Let $s^{\mathcal{F}}_\star$ be a goal decoupled state for which $\langle\rangle$ is not an optimal decoupled plan. Let $\mathcal{A}^{\mathcal{F}}_s$ be a DSSS for $s^{\mathcal{F}}_\star$. Then $\mathcal{A}^{\mathcal{F}}_s$ is safe for $s^{\mathcal{F}}_\star$.*

Note that, if there is no goal-price frontier transition for F^L in $s^{\mathcal{F}}_\star$, then $\emptyset$ is a goal-price frontier set for F^L in $s^{\mathcal{F}}_\star$. If this is so for all F^L on which a goal is defined, then, consequently, $\emptyset$ is a DSSS for $s^{\mathcal{F}}_\star$. This is safe, i.e., we can prune all outgoing transitions of $s^{\mathcal{F}}_\star$, because in this situation the goal price cannot be improved. In other words, goal-price frontier sets encompass a sufficient criterion for safely stopping the search at $s^{\mathcal{F}}_\star$.

5. Special Cases Facilitating More Effective Handling

As we have seen, the construction of safe action sets in general star topologies requires to be rather inclusive, collecting potentially many actions which is detrimental to pruning power. This is specifically so for non-goal decoupled states (and hence most of the search). The main difficulty is the handling of reached leaf conditions p, i.e., reached-enabling sets. As p is already fully reached, in contrast to necessary enabling sets we cannot focus on some small part of p that is still open.

A major remedy is the consideration of more restricted topologies, namely fork and inverted-fork topologies. These are practically relevant, since they can be easily identified and occur regularly in planning domains. The key property of these topologies is *monotonicity* of the pricing function:

- **Positive monotonicity:** *In a fork topology, the price of any one leaf state s^L can only decrease along a search path.*

  This is because the center has no precondition or effect on the leaves, so whenever a leaf path becomes center-compliant (all its preconditions on F^C have been provided), it remains so forever after.

- **Negative monotonicity:** *In an inverted-fork topology, the price of any one leaf state s^L can only increase along a search path.*

  This is because the only cross-factor interaction consists in preconditions of the center on the leaves. So anything the center does can only pose more requirements on the leaves, and whenever a leaf path becomes non-center-compliant, it remains so forever after.

Positive monotonicity is useful because, for reached leaf conditions p, the only future event of interest is one that strictly decreases the price of some leaf state that satisfies p. Hence necessary enablers for such p can now be identified in a more targeted manner, similar to the goal-price frontier sets above. Negative monotonicity is useful because, if a leaf condition p is not reached, then this will remain so and anything requiring p can be pruned.

For the sake of conciseness, we give summary presentations only, deferring some details, in particular the full proofs, to Appendix A. We consider fork topologies in Section 5.1 and inverted-fork topologies in Section 5.2. In Section 5.3, we show how to combine the respective techniques, as special-case treatments for individual fork/inverted-fork leaves as part of general star topologies.

5.1 Forks

As previously hinted, the major ingredient of our treatment for fork topologies is a concept capturing how the price of a reached leaf condition p may be improved. This being the key construction here, we introduce and discuss it in detail. The concept is similar, but not identical to, the definition of goal-price frontier sets:

Definition 9 (Fork-Price Frontier Set)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^\mathcal{F}$ be a decoupled state, let F^L be a leaf factor, and let p be a partial assignment to F^L reached in $s^\mathcal{F}$.*

*We say that an F^L transition $s^L \xrightarrow{a} t^L$ is a **fork-price frontier transition** for p in $s^\mathcal{F}$ if*

(i) s^L is reached in $s^\mathcal{F}$, $\mathsf{prices}[s^\mathcal{F}](s^L) + cost(a) < \mathsf{prices}[s^\mathcal{F}](t^L)$, and $\mathsf{center}[s^\mathcal{F}] \not\models pre(a)[F^C]$; and

(ii) $s^L \xrightarrow{a} t^L$ is part of a simple F^L path π^L from $s_0[F^L]$ to an F^L state r^L where $r^L \models p$ and $cost(\pi^L) < \mathsf{prices}[s^\mathcal{F}](r^L)$.

*An action set A is a **fork-price frontier set** for p in $s^\mathcal{F}$ if, for every path π^L as in (ii), the segment $\pi^L_{s \to r}$ of π^L starting with $s^L \xrightarrow{a} t^L$ contains at least one action from A.*

There are three differences to goal-price frontier sets (Definition 7): (a) the additional requirement in (i) that $\mathsf{center}[s^\mathcal{F}] \not\models pre(a)[F^C]$; (b) that π^L is required to be simple in (ii), i.e., visit each F^L state at most once; (c) the weaker condition $cost(\pi^L) < \mathsf{prices}[s^\mathcal{F}](r^L)$ in (ii). Differences (a) and (b) are sound in fork topologies but not in general, so are benefits of the focus on forks. Difference (c) is due to the difference between non-goal vs. goal decoupled states: whereas on the latter the goal price must be reduced globally, on non-goal decoupled states the price reduction will pertain only to individual leaf states r^L. On goal decoupled states $s^\mathcal{F}_\star$ in fork topologies, we can get rid of difference (c), i.e., instead of $cost(\pi^L) < \mathsf{prices}[s^\mathcal{F}](r^L)$ we can use $cost(\pi^L) < \mathsf{goalprice}[s^\mathcal{F}_\star](F^L)$ like in Definition 7. We refer to this variant as **fork-goal-price frontier sets**.

As before, we show the correctness of our construction, i.e., that it indeed captures all potential enabling actions. The proof is similar to that for goal-price frontier sets in Lemma 3, with simple additions tackling the differences (a) – (c):

Lemma 4. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^\mathcal{F}$ be a solvable decoupled state, let F^L be a leaf factor, and let p be a partial assignment to F^L reached in $s^\mathcal{F}$. Let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}$, and let π be a global plan given $\pi^\mathcal{F}$. Let A be a fork-price frontier set for p in $s^\mathcal{F}$.*

Define a_t, π^{past}, and π^{future} as before. If there exists $k > t$ such that, denoting by $\langle a^L_1, \dots, a^L_i \rangle$ and $s^L_0, \dots, s^L_i$ the F^L actions, respectively states, in π prior to a_k, we have $cost(\langle a^L_1, \dots, a^L_i \rangle) <$ $\mathsf{prices}[s^{\mathcal{F}}](s^L_i)$ and $s^L_i \models p$, then $\{a_t, \dots, a_{k-1}\} \cap A \neq \emptyset$.

Proof (sketch). Similarly to the proof of Lemma 3, we consider the smallest index j such that a^L_j is on π^{future} and $cost(\langle a^L_1, \dots, a^L_j \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s^L_j)$, and we prove that $s^L_{j-1} \xrightarrow{a^L_j} s^L_j$ is a fork-price frontier transition for p in $s^{\mathcal{F}}$, from which the claim follows directly.

Difference (c) does not change anything as it is merely a different condition on the F^L path end state, s^L_i in our case here, provided by prerequisite. It remains to prove that (a) $\mathsf{center}[s^{\mathcal{F}}] \not\models pre(a^L_j)[F^C]$ and (b) π^L is simple.

Regarding (a), as a^L_j affects F^L, in a fork topology a^L_j cannot affect the center. Further more, from $\mathsf{prices}[s^{\mathcal{F}}](s^L_{j-1}) + cost(a^L_j) < \mathsf{prices}[s^{\mathcal{F}}](s^L_j)$ which is proved exactly as in Lemma 3, we can therefore conclude that a^L_j cannot be reached in $s^{\mathcal{F}}$. We do know, however, that s^L_{j-1} is reached in $s^{\mathcal{F}}$. As the only non-F^L precondition of a^L_j must be on F^C, (a) follows.

Regarding (b), in a fork topology, any cheapest compliant leaf path is simple, because without center preconditions nor effect on the leaf there is no reason to visit the same leaf state twice. □

A similar claim holds for fork-goal-price frontier sets A, i.e., if it holds that $cost(\pi^L) <$ $\mathsf{goalprice}[s^{\mathcal{F}}_\star](F^L)$, then π^{future} contains an action from A. The proof is a straightforward combination of those for Lemmas 3 and 4, essentially applying arguments (a) and (b) from the latter as an extension of the former to tackle the definition differences (a) and (b) explained above.

From a practical perspective, in difference to goal-price frontier sets, item (ii) in Definition 9 is moot. First, we cannot exploit (b) that π^L must be simple, as the question whether there exist vertex-disjoint paths from $s_0[F^L]$ to s^L and from t^L to r^L is the 2-vertex-disjoint paths problem for directed graphs, which is known to be **NP**-complete (Fortune, Hopcroft, & Wyllie, 1980). Second, in the weaker condition (c) $cost(\pi^L) < \mathsf{prices}[s^{\mathcal{F}}](r^L)$, the value $\mathsf{prices}[s^{\mathcal{F}}](r^L)$ we're comparing the cost of π^L to is not a constant, i.e., depends on π^L. One could check the condition through a search below t^L, or through precomputing all-pairs shortest paths on the leaf state spaces, but given the associated overhead neither is promising. So we settle for the naïve approximation collecting all actions a labeling the fork-price frontier transitions $s^L \xrightarrow{a} t^L$. For fork-goal-price frontier sets, of course, we can apply the same filter as before, i.e., collect a only where $g^*(s^L) + cost(a) + h^*(t^L) <$ $\mathsf{goalprice}[s^{\mathcal{F}}_\star](F^L)$.

We are now ready to define decoupled strong stubborn sets for fork topologies. For conciseness, we subsume both, the non-goal and goal cases, into a single definition:

Definition 10 (DSSS: Forks)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^{\mathcal{F}}$ be a decoupled state.*

*An action set $\mathcal{A}^{\mathcal{F}}_s$ is a **fork-decoupled strong stubborn set (FDSSS)** for $s^{\mathcal{F}}$ if all of the following conditions hold:*

- *(i) If $s_\star$ is not reached in $s^{\mathcal{F}}$, then $\mathcal{A}^{\mathcal{F}}_s$ contains a decoupled necessary enabling set for $s_\star$ in $s^{\mathcal{F}}$. If $s_\star$ is reached in $s^{\mathcal{F}}$, then, for every F^L where $s_\star[F^L] \neq \emptyset$, $\mathcal{A}^{\mathcal{F}}_s$ contains a fork-goal-price frontier set for F^L in $s^{\mathcal{F}}_\star$.*
- *(ii) For all actions $a \in \mathcal{A}^{\mathcal{F}}_s$ not reached in $s^{\mathcal{F}}$, $\mathcal{A}^{\mathcal{F}}_s$ contains a decoupled necessary enabling set for $pre(a)$ in $s^{\mathcal{F}}$.*

(iii) For all center actions $a^C \in \mathcal{A}_s^{\mathcal{F}} \cap \mathcal{A}^C$ reached in $s^{\mathcal{F}}$, $\mathcal{A}_s^{\mathcal{F}}$ contains all actions a' interfering with a^C and where $pre(a^C) \parallel pre(a')$.

(iv) For all leaf actions $a^L \in \mathcal{A}_s^{\mathcal{F}} \cap \mathcal{A}^L[F^L]$ reached in $s^{\mathcal{F}}$, $\mathcal{A}_s^{\mathcal{F}}$ contains a fork-price frontier set for $pre(a^L)[F^L]$ in $s^{\mathcal{F}}$.

Recall here that, for fork and inverted-fork topologies, the center actions $\mathcal{A}^C$ and the leaf actions $\mathcal{A}^L[F^L]$ are disjoint, $\mathcal{A}^C \cap \mathcal{A}^L[F^L] = \emptyset$. Item (i) is an obvious combination handling non-goal $s^{\mathcal{F}}$ like Definition 6, and handling goal $s^{\mathcal{F}}$ like Definition 8 but with fork-goal-price frontier sets. Item (ii) is as before, item (iii) as well except that, as it turns out, we need to select interfering actions only for center actions $a^C \in \mathcal{A}_s^{\mathcal{F}} \cap \mathcal{A}^C$. Item (iv) replaces reached-enabling sets with fork-price frontier sets as advertised, doing so only for leaf actions as center actions do not have leaf preconditions in a fork.[6]

As we mentioned before, our safety proof makes use of a notion of past-maximality different from that used for star topologies (Definition 3). We defer the details to Appendix A. Essentially, π is **fork-past-maximal** if, whenever on π^{future} we apply a leaf action a_k to a leaf state s_i^L already reached in $s^{\mathcal{F}}$, then we do so on a leaf path achieving s_i^L at a price cheaper than that in $s^{\mathcal{F}}$. Intuitively, this makes sense as, otherwise, we could have just as well used a compliant path for s_i^L in π^{past} already. Formally, every global plan can be transformed into a fork-past-maximal global plan of equal of cheaper cost.

Theorem 3. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^{\mathcal{F}}$ be a solvable decoupled state for which $\langle\rangle$ is not an optimal decoupled plan, and let $\pi^{\mathcal{F}}$ be a strongly optimal decoupled plan for $s^{\mathcal{F}}$. Let $\mathcal{A}_s^{\mathcal{F}}$ be an FDSSS for $s^{\mathcal{F}}$.*

Then $\mathcal{A}_s^{\mathcal{F}}$ contains a center action starting a permutation of $\pi^{\mathcal{F}}$.

Proof (sketch). The proof is similar to that of Theorems 1 and 2. There is one major difference.

As before, we consider a global plan π for $\pi^{\mathcal{F}}$. We assume without loss of generality that π is fork-past-maximal. We denote a_t, π^{past}, and π^{future} as before.

The argument (a) that there must be an action a shared between $\mathcal{A}_s^{\mathcal{F}}$ and π^{future} is the same as in the proof of Theorem 1 in the case of non-goal $s^{\mathcal{F}}$. For goal $s^{\mathcal{F}}$, it is the same as in the proof of Theorem 2, replacing Lemma 3 with the variant of Lemma 4 for fork-goal-price frontier sets.

Denoting by a_k the first shared action, that (b) a_k is reached in $s^{\mathcal{F}}$ follows with the same argument as in the proof of Theorem 1.

The one major difference is that we have to prove next that *(c) a_k is a center action*, because Definition 10 (iii) includes interfering actions for center actions $a^C \in \mathcal{A}_s^{\mathcal{F}} \cap \mathcal{A}^C$ only.

Once (c) is shown, the same argument as in Theorem 1 (c) shows that a_k does not interfere with any of the actions a_i, $t \leq i \leq k-1$. The same argument as in Theorem 1 (e) shows that a_k can be moved to the start of π^{future}. The same concluding argument as in Theorem 1 shows the claim.

So, how do we show (c)? If a_k is a leaf action, $a_k \in \mathcal{A}^L[F^L]$, then by Definition 10 (iv) $\mathcal{A}_s^{\mathcal{F}}$ contains a fork-price frontier set A for $p := pre(a_k)[F^L]$ in $s^{\mathcal{F}}$. As a_t is a center action, given the fork topology we know that $a_k \neq a_t$ and hence $k > t$. Hence fork-past-maximality applies, showing that $cost(\langle a_1^L, \ldots, a_i^L \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s_i^L)$ where $\langle a_1^L, \ldots, a_i^L \rangle$ and $s_0^L, \ldots, s_i^L$ denote the

6. We remark that, in the latter, a fork-price frontier set needs to be collected even if $pre(a^L)[F^L] = \emptyset$. This is because a^L may need to be on π^{future} to correct a detrimental effect of some other, not-yet-reached, leaf action on π^{future}. In the special case where F^L consists of a single variable though, the latter cannot happen, so in that special case a fork-price frontier set is not required for $pre(a^L)[F^L] = \emptyset$. We exploit that special case in our implementation.

F^L actions, respectively states, in π prior to a_k. Applying Lemma 4 yields that $\{a_t, \dots, a_{k-1}\} \cap A \neq \emptyset$, in contradiction to a_k being the first shared action, concluding the proof. □

Corollary 3. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^\mathcal{F}$ be a solvable decoupled state for which $\langle\rangle$ is not an optimal decoupled plan. Let $\mathcal{A}_s^\mathcal{F}$ be an FDSSS in $s^\mathcal{F}$. Then $\mathcal{A}_s^\mathcal{F}$ is safe for $s^\mathcal{F}$.*

We remark that the definition of fork-topology DSSS in the authors' previous IJCAI'16 paper on this topic (Gnad et al., 2016) does not contain item (iv). This is incorrect, i.e., does *not* guarantee safety. It may indeed forsake completeness, i.e., prune all decoupled plans for $s^\mathcal{F}$, because even if a leaf action a^L's precondition is already reached at the best possible price in $s^\mathcal{F}$, the price of the actual leaf state s^L we must apply a^L to may not yet be reached at all. Example 3 in Appendix A shows a case where this happens. During our work on the IJCAI'16 paper, the intricacies of enabling reached leaf conditions unfortunately escaped our attention. Empirically, our corrected version of fork-topology DSSS does not lose a lot of pruning power as we shall see in Section 7.2.

5.2 Inverted Forks

Our modifications for inverted-fork topologies are much simpler than those for forks. This has two main reasons:

- On goal decoupled state $s_\star^\mathcal{F}$, we don't need to do anything at all. This is because, with negative monotonicity – pricing functions increasing monotonically along search paths – the prices below $s_\star^\mathcal{F}$ can never improve. So there is no need to search below $s_\star^\mathcal{F}$, and we don't need to define any pruning techniques for that case (in other words: just set $\mathcal{A}_s^\mathcal{F} := \emptyset$).
- On non-goal decoupled states, the intricacies of enabling reached leaf conditions p remain essentially the same as in general star toplogies – all we know is that p is false after π^{past} but may need to be made true in the future. Hence we need to include reached-enabling sets just like for general star topologies.

Given this, our modifications are confined to discarding any actions a from $\mathcal{A}_s^\mathcal{F}$ whose leaf preconditions aren't reached in $s^\mathcal{F}$ – which preserves safety simply because such a can never occur on π^{future}. For the sake of completeness, let us state the latter formally:

Proposition 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ an inverted-fork factoring. Let $s^\mathcal{F}$ be a solvable decoupled state. Let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}$, and let π be a global plan given $\pi^\mathcal{F}$. Let a be an action where $pre(a)[\mathcal{V} \setminus F^C]$ is not reached in $s^\mathcal{F}$.*

Define a_t, π^{past}, and π^{future} as before. Then a does not occur on π^{future}.

Proof. Assume to the contrary that a does occur on π^{future}. There must be at least one leaf factor where the precondition of a on that factor is not reached in $s^\mathcal{F}$. Let F^L be such a leaf factor. Then $pre(a)[F^L]$ must be supported, in π, by a $\pi^C[s^\mathcal{F}] \circ \pi^C[\pi^\mathcal{F}]$-compliant F^L path π^L, ending in s^L where $s^L \models pre(a)[F^L]$. Given the inverted-fork structure, the actions in π^L have preconditions on F^L only. Therefore, π^L also is $\pi^C[s^\mathcal{F}]$-compliant. But then, $\mathsf{prices}[s^\mathcal{F}](s^L) < \infty$ in contradiction. □

The simplified definitions are as follows. First, necessary enabling sets need to worry about the center only, as an unreached leaf condition is unsolvable; and actions whose leaf preconditions are not reached can be discarded:

Definition 11 (Center-Necessary Enabling Set)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ an inverted-fork factoring. Let $s^\mathcal{F}$ be a decoupled state, and let p be a partial variable assignment to F^C not reached in $s^\mathcal{F}$.*

*An action set A is a **center-necessary enabling set** for p in $s^\mathcal{F}$ if there exists $v \in vars(p)$ such that $\mathsf{center}[s^\mathcal{F}][v] \neq p[v]$, and $A = \{a \in \mathcal{A} \mid \mathit{eff}(a)[v] = p[v], pre(a)[\mathcal{V} \setminus F^C]$ is reached in $s^\mathcal{F}\}$.*

For reached-enabling sets, nothing changes. This is because, cf. Definition 5, these catch actions from leaf paths π^L starting in reached-enabling states s^L, one of whose properties is that s^L is reached in $s^\mathcal{F}$. But then, as an F^L leaf path has preconditions only on F^L itself, all actions on π^L are already reached, and there are no unreached leaf preconditions to prune.

The definition of DSSS now mirrors that of DSSS for general star topologies:

Definition 12 (DSSS: Inverted Forks)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ an inverted-fork factoring. Let $s^\mathcal{F}$ be a non-goal decoupled state where $s_\star[\mathcal{V} \setminus F^C]$ is reached in $s^\mathcal{F}$.*

*An action set $\mathcal{A}_s^\mathcal{F}$ is an **inverted-fork decoupled strong stubborn set (IFDSSS)** for $s^\mathcal{F}$ if all of the following conditions hold:*

(i) $\mathcal{A}_s^\mathcal{F}$ contains a center-necessary enabling set for $s_\star[F^C]$ in $s^\mathcal{F}$.

(ii) For all center actions $a \in \mathcal{A}_s^\mathcal{F}$ not reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains a center-necessary enabling set for $pre(a)[F^C]$ in $s^\mathcal{F}$.

(iii) For all actions $a \in \mathcal{A}_s^\mathcal{F}$ reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains all actions a' interfering with a where $pre(a) \parallel pre(a')$ and $pre(a')[\mathcal{V} \setminus F^C]$ is reached in $s^\mathcal{F}$.

(iv) For all actions $a \in \mathcal{A}_s^\mathcal{F}$ reached in $s^\mathcal{F}$, and for all F^L where $pre(a)[F^L] \neq \emptyset$, $\mathcal{A}_s^\mathcal{F}$ contains a reached-enabling set for $pre(a)[F^L]$ in $s^\mathcal{F}$.

Compared to DSSS for general star topologies (Definition 6), we can restrict focus to decoupled states $s^\mathcal{F}$ where the leaf goal is reached, because otherwise $s^\mathcal{F}$ is unsolvable. We use center-necessary enabling sets instead of decoupled necessary enabling sets. We discard actions whose leaf preconditions are not reached, and given this, unreached leaf actions are never included, so only unreached center actions need to be supported.

Safety of this construction follows immediately from safety of DSSS for general star topologies, together with Proposition 1:

Corollary 4. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ an inverted-fork factoring. Let $s^\mathcal{F}$ be a solvable non-goal decoupled state. Let $\mathcal{A}_s^\mathcal{F}$ be an IFDSSS in $s^\mathcal{F}$. Then $\mathcal{A}_s^\mathcal{F}$ is safe for $s^\mathcal{F}$.*

Proof. The proof of Theorem 1 remains intact almost exactly as stated. The only addition we need to make is in arguments (a) – (c), where we invoke Proposition 1 to show that the shared action identified on π^{future} (a,b), respectively the actions preceding a_k on π^{future} (c), do not have unreached leaf preconditions. □

We remark that, in the special case of inverted-fork topologies where all leaf factors consist of a single state variable only, one can modify Definition 12, using only center actions in item (iii), and replacing item (iv) with a rule that includes center actions a' whose leaf preconditions compete with those of a center action a already included into $\mathcal{A}_s^\mathcal{F}$. We use this more targeted definition as an optimization in our implementation. Formal details are provided in Definition 15 in Appendix A. Although this special case is interesting from a theoretical point of view, we did not observe a major impact in most of the planning domains that we ran our analysis on, as we shall see in Section 7.

5.3 Fork/Inverted-Fork Leaves in General Star Topologies

One can also combine the observations just made for fork and inverted-fork topologies, to obtain an enhanced variant of DSSS for general star topologies, exploiting the properties of individual fork/inverted-fork leaves. This is important as, in practice, it often happens that even in general star topologies there are (inverted-)fork leaves that can be treated more efficiently by the special-case algorithms.

Recall that fork/inverted-fork leaves behave like the leaves in the respective structure. That is, a fork leaf is one on which the center has neither precondition nor effect; and an inverted-fork leaf is one on which the center has no effect, and that has no precondition on the center. If such leaves occur in a general star topology, then we don't get positive/negative monotonicity at the global level, but we do get it for each of these individual leaves. This can be exploited in the same manner as specified above.

We define **enhanced decoupled strong stubborn sets (EDSSS)** towards this end. As the details are simple yet cumbersome to spell out formally, we defer them to Appendix A. Here is a summary:

(i) *Goal enablers:* For non-goal $s^{\mathcal{F}}$, use decoupled necessary enabling sets. For goal $s^{\mathcal{F}}$, use goal-price frontier sets for non-fork leaves, and use fork-goal-price frontier sets for fork leaves. Discard actions with unreached inverted-fork preconditions.

(ii) *Unreached action enablers:* Use decoupled necessary enabling sets. Discard actions with unreached inverted-fork preconditions.

(iii) *Reached action interference:* As before, but only for non-fork-leaf actions, and discarding actions with unreached inverted-fork preconditions.

(iv) *Reached action enablers:* For non-fork-leaf actions, use reached-enabling sets. For fork-leaf actions, use fork-price frontier sets.

The proof of safety mirrors that of Theorem 1, the only major addition being a new proof item showing that a_k must be a non-fork action. The argument used for that purpose is the same one used in Theorem 3 to prove that a_k must be a center action. The notions of past-maximality and fork-past-maximality are combined in these arguments, in the sense of assuming these properties for the individual leaves F^L as appropriate.

6. Exponential Separations

Before proceeding to the empirical part of our research, let us state some basic theoretical facts evaluating the power of DSSS. We only give proof sketches in this section; full proofs are given in Appendix A.2. We say that a search space reduction method X is *exponentially separated* from a method Y if there exists a parameterized example family F such that, on F, X yields an exponentially stronger reduction than Y.

Decoupled search and SSS are complementary in that each is exponentially separated from the other:

Theorem 4. *Decoupled search is exponentially separated from SSS, and vice versa.*

Our running example with locations A and B is a suitable family F for the first claim. There are only 3 reachable decoupled states ($s_0^{\mathcal{F}}$; drive to B; drive back). But SSS do not yield any pruning because, in any state s, to make progress to the goal, $\mathcal{A}_s$ must include an applicable *(un)load* action; which interferes with the applicable *drive* action; which in turn interferes with all applicable *(un)load* actions. The opposite claim follows from an example shown in the proof of Lemma 1 in Gnad and Hoffmann (2019). Here, the state space under SSS pruning is linear; decoupled search, however, cannot exploit the independence because of leaf-leaf interactions that forbid a star factoring, but never trigger during search.

Trivially, DSSS is exponentially separated from each of decoupled search and SSS, simply because DSSS naturally generalizes each of these components, so we can use the same families F as in Theorem 4. As a much stronger testimony to the power of DSSS, there are cases where it is exponentially separated from *both* its components:

Theorem 5. *There exists a parameterized example family F such that, on F, DSSS yields an exponentially stronger reduction than both, decoupled search and SSS.*

Two suitable families F arise from simple modifications of our running example. First, say we have M trucks and $N * M$ packages, where each truck t_i is associated with a group of N packages that only t_i can transport. The number of reachable decoupled states is exponential in M because all trucks must be in the center factor. The SSS-pruned reachable standard state space has size exponential in N because including an *(un)load* action into $\mathcal{A}_s$ necessitates, due to interference via the truck move as above, to include *all* applicable *(un)load* actions for the respective package group. However, *in decoupled search with DSSS pruning, there are only $M + 1$ reachable states*. This is because the two sources of pruning power combine gracefully. Decoupling gets rid of the blow-up in N (the packages within a group become independent leaves), while DSSS gets rid of the blow-up in M (only a single truck is committed to at a time).

In our second example, DSSS even is exponentially *more* than the sum of its components: stubborn sets have exponentially more impact on the decoupled search space than on the standard one. Say we have N packages and M trucks (where every truck may transport every package). Then decoupled search blows up in M, and SSS does not do anything because any package may require any truck. Applying DSSS to decoupled search, no truck move is pruned in $s_0^{\mathcal{F}}$. However, after applying any one *drive*(t_i, A, B) action, all package prices are the cheapest possible ones, the goal-price frontier is empty, and DSSS stops the search. So, again, there are only $M+1$ reachable states. As we shall see next, similar phenomena seem to occur in the standard IPC Logistics benchmarks.

7. Experiments

We focus our experimental evaluation on optimal planning and proving unsolvability of planning instances. These algorithmic problems are the ones where partial-order reduction techniques have typically been applied to in the past, because of the need to explore large state spaces exhaustively. Here, strong stubborn sets pruning can exponentially reduce the search effort, facilitating the problem significantly. We compare the performance of decoupled strong stubborn sets pruning (DS^3) to its base components, i.e., standard search with strong stubborn sets pruning (S^3), decoupled search (DS), as well as standard search without pruning (B).

We start by describing the setup and our implementation in Section 7.1, and then discuss the experimental evaluation in Sections 7.2 (optimal planning) and 7.3 (proving unsolvability).

7.1 Experimental Setup & Implementation

All experiments are conducted on a cluster of Intel E5-2660 machines running at 2.20 GHz, with a time cut-off of 30 minutes and a memory limit of 4 GB. Our implementation extends the decoupled search planner of Gnad and Hoffmann (2018), which is based on the Fast Downward planning system (FD) (Helmert, 2006). We use the Lab software package to execute the experiments (Seipp, Pommerening, Sievers, & Helmert, 2017).

The decoupled search implementation supports several factoring strategies that can identify different topologies, among others fork and inverted-fork factorings (Gnad et al., 2015), and also more general star topologies (Gnad et al., 2017a). We employ the fork and inverted-fork factoring strategies to illustrate the advantages of the special case optimizations developed in Section 5, using the improved factoring strategies from Gnad et al. (2017a). To evaluate our general framework, we choose the incident-arcs strategy as introduced in Gnad et al. (2017a), which is able to generate star factorings in the highest number of instances in the benchmark set that we use. The strategy computes so-called **strict-star** factorings, which allow for arbitrary interaction between center and leaves, but no interaction across leaves. This restricts the possible factorings $\mathcal{F}$ as defined in Section 3.1 in that center actions affecting a leaf factor F^L, must have preconditions and effects only on F^L and F^C. More formally, let a^C be a center action, if there exists an $F^L \in \mathcal{F}$ such that $vars(\mathit{eff}(a^C)) \cap F^L \neq \emptyset$ then $vars(pre(a^C)) \cup vars(\mathit{eff}(a^C)) \subseteq F^L \cup F^C$. This prohibits center actions that (1) affect multiple leaf factors, or (2) have an effect on one and are preconditioned by another leaf.

The three factoring strategies that we use are typically very complementary, identifying valid factorings on different planning instances. Therefore, we show results for each of the strategies separately, on benchmarks in which the strategy is successful. Like previous work on decoupled search, we **abstain** from tackling an instance whenever the resulting factoring has less than two leaf factors. The rationale behind this is that the potential gain of decoupled search over standard search is exponential in the number of leaves. We remark that the factoring process is very fast, terminating within 1s in almost all instances. Therefore, when no proper factoring could be found, we can still invoke standard search without loosing a lot of time. We do *not* do this to focus our evaluation on benchmark instances where decoupled search is actually performed.

The definitions in Section 4 do not fully specify *how* to compute decoupled strong stubborn sets, but rather state the properties such sets need to have. In particular, there are several ways to compute (1) decoupled necessary enabling sets, (2) reached-enabling sets, and (3) goal-price frontier sets. For (1), we stick to the ordering given in Definition 4, namely we first choose unreached leaf variable assignments, then unreached center assignments. If this still does not result in a unique choice, we stick to the FD variable-ordering, which has also been used in prior work on strong stubborn sets (Wehrle & Helmert, 2014). In combination with our use of action interference, where only interfering actions with *agreeing* preconditions are added to the DSSS, our selection strategy corresponds to the "full/mutex + FD" strategy of Wehrle and Helmert (2014). For the standard search variant of strong stubborn sets pruning that we compare to (S^3), we use the strategy "full-syntactic-SSS-EC", which is the strongest variant of SSS pruning that is implemented in a recent Fast Downward. Regarding (2) and (3), we use the approximations as described in Section 4.2 and Section 4.3, respectively.

The special case definitions from Section 5 also leave some freedom, namely in how to compute fork(-goal)-price frontier sets and center-necessary enabling sets. For the latter, we stick to

the aforementioned static selection of yet unreached assignments. Regarding the former, we implemented the algorithms outlined in Section 5.1, except that we do not check if a leaf path π^L is simple.

We show results for two different configurations of decoupled strong stubborn sets, the plain variant implementing the definitions from Section 4 (DS^3), and a variant that uses all optimizations from Section 5 whenever they are applicable (DS^3_O).

As a preview on the empirical results, it turns out that just like in standard search (Alkhazraji et al., 2012; Wehrle et al., 2013; Wehrle & Helmert, 2014), there exist domains where stubborn sets pruning does not result in any reduction of the search space size. Yet, computing a (decoupled) strong stubborn set for every state expanded in the search incurs a significant runtime overhead. Therefore, we implement a simple **safety belt** mechanism to disable the pruning in planning instances where stubborn sets are not effective. To do so, we switch the stubborn sets off if after the first 1000 expanded states less than 1% of the transitions have been pruned (a similar mechanism has been used in Torralba and Hoffmann (2015)). We keep the safety belt rather permissive to only disable the pruning in instances where effectively no transitions are being pruned. This allows for a detailed analysis on how much pruning is needed to make up for the computational overhead of computing a (decoupled) strong stubborn set in every search state. Expanding 1000 states usually takes only little time, so we expect the total overhead to be small if the pruning does not pay off. On the other hand, if only a small fraction of the transitions could be pruned during the first 1000 expansions, chances are high that it remains like this throughout the entire search. We shall see in the next two sections that both conjectures are valid in almost all domains.

7.2 Optimal Planning

We evaluate our algorithms in optimal planning on all STRIPS benchmarks from the optimal tracks of the International Planning Competition (IPC) ('98-'18). To analyze the pruning power of the strong stubborn sets, we show results for blind search. Additionally, we run A* search with the $h^{\text{LM-cut}}$ heuristic (Helmert & Domshlak, 2009), a well-performing configuration for optimal planning. When using $h^{\text{LM-cut}}$, we include Complementary 2 (C2) to the comparison, the best non-portfolio planner in the optimal track of IPC'18 (Franco, Lelis, & Barley, 2018). Complementary is based on symbolic pattern databases (Franco, Torralba, Lelis, & Barley, 2017).

As a general remark, observe that, since in decoupled search the pruning only happens over *center* actions, the potential for pruning is significantly reduced compared to standard search. No reduction within the leaves can be achieved, given the distributed nature of decoupled search. Furthermore, the necessity to include reached-enabling sets (in the star case) and fork-price frontier sets (for fork factorings) reduces the pruning power in the decoupled setting even more. On the other hand, the node generation time is tremendously higher in decoupled search, due to the maintenance of the pricing function. Therefore, a smaller search space reduction can already lead to a speed-up over pure decoupled search, because the relative overhead of computing a strong stubborn set for each expanded state is smaller.

We start with the more general star factorings obtained from the incident-arcs strategy – Table 1 shows the results for blind search. The left part shows coverage data – number of instances solved – for the 7 competitors. Observe that, as expected, DS and S^3 are very orthogonal. While DS works well on domains with a pronounced star topology, e.g. Logistics or NoMystery, S^3 performs well in domains that have many permutable action sequences, e.g. ParcPrinter or Woodworking. The

	Coverage								Pruning Power								
							+safety b.		Transitions			Search Space			Runtime		
Domain	#	B	S^3	DS	DS^3	DS^3_O	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O
Depots	22	4	4	4	4	4	4	4	4	3	0	4	1.0	1.0	3	0.1	0.3
Driverlog	20	7	7	**11**	**11**	**11**	7	**11**	7	24	1	7	1.0	1.0	6	0.3	0.4
Elevators08	30	**14**	12	9	9	9	12	9	9	17	0	9	1.1	1.0	9	0.2	0.6
Elevators11	20	**12**	10	7	7	7	10	7	7	19	0	7	1.1	1.0	7	0.2	0.6
Floortile11	20	**2**	**2**	1	1	1	**2**	1	1	1	0	1	1.0	1.0	1	0.2	0.8
Logistics00	28	10	10	22	22	**23**	10	**23**	10	32	55	10	1.0	2.0	8	0.4	1.0
Logistics98	35	2	2	2	2	2	2	2	2	39	0	2	1.2	1.0	2	0.4	0.7
Maintenance	5	5	5	5	5	5	5	5	5	85	76	5	3513.7	1747.9	2	4838.5	5893.4
Miconic	145	**50**	40	36	36	36	**50**	36	36	0	2	36	1.0	1.0	21	0.0	1.1
Mprime	6	**6**	**6**	4	4	4	**6**	4	4	0	13	4	1.0	1.0	3	0.0	0.2
Mystery	1	1	1	1	1	1	1	1	1	0	12	1	1.0	1.0	0		
NoMystery	20	8	8	**17**	13	13	8	**17**	8	20	0	8	1.2	1.0	6	0.1	0.0
Openstacks08	30	**22**	20	19	19	19	21	19	19	8	0	19	1.3	1.0	17	0.1	0.4
Openstacks11	20	**17**	15	14	14	14	16	14	14	9	0	14	1.3	1.0	14	0.1	0.3
Openstacks14	20	**3**	**3**	2	1	1	**3**	2	1	3	0	1	1.1	1.0	1	0.0	0.3
Organic-Split	16	**6**	5	3	3	2	**6**	2	2	20	21	2	1.8	1.8	2	0.1	0.0
ParcPrinter08	13	1	**13**	3	6	6	**13**	6	6	63	25	1	60.0	1.2	0		
ParcPrinter11	10	0	**10**	2	5	5	**10**	5	5	64	25	0			0		
Pathways	30	4	4	4	4	4	4	4	4	53	27	4	29.3	2.4	2	25.2	1.7
PSR	48	**47**	**47**	46	46	46	**47**	46	46	13	17	46	1.1	1.1	11	0.1	1.3
Rovers	40	6	7	7	**8**	**8**	7	**8**	7	22	23	6	1.6	1.5	2	0.7	1.6
Satellite	36	**6**	**6**	5	5	5	**6**	5	5	41	26	5	190.3	5.1	3	36.1	7.5
Scanalyzer08	9	**6**	3	3	3	3	**6**	3	3	0	2	3	1.0	1.0	0		
Scanalyzer11	5	**4**	1	1	1	1	**4**	1	1	0	2	1	1.0	1.0	0		
Tidybot11	20	**13**	7	**13**	5	5	9	**13**	5	24	20	5	1.2	1.0	5	0.0	0.0
TPP	29	5	5	5	5	5	5	5	5	0	6	5	1.0	1.0	2	0.3	0.6
Transport08	12	9	9	9	9	9	9	9	9	1	0	9	1.0	1.0	6	0.1	0.3
Transport14	17	**6**	4	4	4	4	5	4	4	1	0	4	1.0	1.0	4	0.0	0.3
Trucks	27	**5**	4	4	4	4	**5**	4	3	14	5	3	1.0	1.0	3	0.1	0.6
Woodworking08	26	7	**13**	7	10	10	**13**	10	9	77	57	7	1046.4	180.3	5	358.5	124.2
Woodworking11	17	2	**7**	2	5	5	**7**	5	4	83	64	2	1180.2	203.2	2	406.6	141.3
Zenotravel	20	**8**	7	7	7	7	7	7	7	6	10	7	1.1	1.0	5	0.1	0.9
Others	187	**43**	30	36	27	27	39	36	27	0	0	27	1.0	1.0	22	0.0	0.2
$\sum$	984	341	327	315	306	306	**359**	328									

Table 1: Coverage data (number of instances solved) and pruning power when using blind search with the incident-arcs factoring strategy. "Transitions" shows the average percentage of transitions pruned during search on instances commonly solved by S^3 and DS^3_O. "Search space" and "runtime" columns show the average improvement factors compared to the respective baseline, i.e., B as baseline for S^3 and DS for DS^3_O. We use the number of expanded states until the last f-layer and FD's "total time" on commonly solved instances to measure the improvement (ignoring instances commonly solved in less than $0.1s$ for the runtime factors). "Others" summarizes all domains in which the percentage of pruned transitions was rounded to 0% for both S^3 and DS^3_O. Best coverage is highlighted in **bold** face.

main purpose of this work is to inherit the strength of the stubborn sets where S^3 improves over B. Where S^3 performs better than DS, we want to narrow this gap, ideally making up for the advantage completely, or even doing better than both baselines. As we see from the results, this is really the case – where S^3 improves coverage over B, we also get an improvement for the two decoupled variants with pruning over DS (ParcPrinter, Rovers, and Woodworking). And, indeed, there are two domains where the combination is more than the sum of its components, namely in Logistics00

and Rovers. Yet, DSSS cannot always fully make up for the advantage of S^3. In ParcPrinter, for example, although the coverage increases, DS^3 and DS^3_O still solve only 11 instances compared to 23 of S^3. On the other hand, if there is only little pruning and the overhead does not pay off for S^3, this is also often the case for DS^3 over DS, e.g. in Tidybot.

Comparing the two decoupled strong stubborn sets variants with and without the optimizations from Section 5, it turns out that there is almost no difference. In terms of coverage, there is one Organic-Split instance solved in $1716s$ by DS^3, but not by DS^3_O, which can be attributed to noise. The additional instance solved in Logistics00, however, is indeed due to an increased pruning power of the special case handling of the fork factoring in this domain. When taking a more detailed look at the amount of pruning, we observe no significant difference when enabling the optimizations in most domains. DS^3_O prunes more transitions in 13 domains, leading to a smaller search space in one instance in each of Woodworking08 and Zenotravel, and in all instances of Logistics00. In the latter, the average percentage of pruned transitions increases from 17% to 46% on instances commonly solved by DS^3 and DS^3_O. The runtime in these instances decreases by up to one order of magnitude, but they are all solved within $10s$, anyway. In the other domains, the additional pruning is mostly due to pruning on decoupled goal states, which has only a minor effect on the overall search. Because of this almost identical behaviour (except for one domain), we only report the pruning power data for DS^3_O.

Enabling the safety belt leads to superior coverage for both S^3 and DS^3_O. DS^3_O with safety belt almost fully makes up for the instances that without safety belt are lost compared to DS (there is one exception in Organic-Split). Except in this instance, the safety belt, even in its permissive usage where the pruning is only disabled if it basically does not remove any transition, results in coverage that is always the maximum of DS and DS^3_O without safety belt. The result is not as good for S^3 with safety belt, which loses 16 instances compared to the best of B and S^3.

In terms of total coverage, enabling strong stubborn set pruning in both standard and decoupled search results in a worse performance. Although the coverage increases in domains where stubborn sets reduce the size of the search space, the overhead in the other domains outweighs this advantage. With the safety belt mechanism, one can reliably get the best of both worlds, i.e., use the pruning where it is useful and disable it when it does not lead to any search space reduction.

Let us take a closer look at the pruning power. The "Transitions" columns show the percentage of transitions pruned during the search, on instances commonly solved by S^3 and DS^3_O. The improvement factors for search space size and runtime are ratios over the respective per-domain sum of B over S^3 for S^3, and DS over DS^3_O for DS^3_O. So we always compare to the baseline variant without the pruning, *not* to the common baseline B. Observe that for stubborn sets to be beneficial, a significant percentage of the transitions needs to be pruned. An extreme case is Tidybot, where even with 20% of the transitions pruned the size of the search space does not change for decoupled search and only slightly for standard search, and the runtime overhead is prohibitive. More expectedly, this effect also becomes visible in the "Others" domains, where the average transition reduction is rounded to 0% for S^3 and DS^3_O. Typically, when S^3 achieves a substantial pruning, this is also true for DS^3_O, although the reduction is almost always less pronounced. On the positive side, the slowdown is usually smaller, too, in domains with little pruning. In some domains, DS^3_O even prunes more transitions than S^3 (in percent), and this translates into a higher runtime benefit. This happens, e.g., in Logistics00, PSR, Rovers, and Zenotravel.

Overall, decoupled strong stubborns sets pruning results in improved runtime (coverage) in 8 (4) of the 31 domains (counting domains that appeared in different competitions only once), in-

	Coverage									Pruning Power								
							+safety b.			Transitions			Search Space			Runtime		
Domain	#	B	S^3	DS	DS^3	DS^3_O	S^3	DS^3_O	C2	#	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O
DataNetworks	20	12	12	12	12	12	12	12	**14**	12	0	5	12	1.0	1.0	11	0.9	0.9
Depots	22	7	7	7	7	7	7	7	7	7	3	1	7	1.0	1.0	5	0.9	0.9
Driverlog	20	13	**14**	13	13	13	**14**	13	**14**	13	16	3	13	1.1	1.0	8	1.0	0.9
Elevators08	30	22	22	22	22	22	22	22	**25**	22	23	1	22	1.2	1.0	21	1.5	0.8
Elevators11	20	18	18	18	18	18	18	18	**19**	18	24	2	18	1.1	1.0	18	1.2	0.8
Logistics00	28	20	21	**28**	**28**	**28**	21	**28**	22	21	28	45	20	3.4	1.6	10	4.2	1.8
Logistics98	35	6	6	**7**	6	**7**	6	**7**	6	6	38	2	6	2.3	1.0	3	2.8	0.9
Maintenance	5	5	5	5	5	5	5	5	5	5	77	72	5	1.0	1.0	0		
Miconic	145	**136**	**136**	135	135	135	**136**	135	99	135	0	3	135	1.0	1.0	91	0.8	0.9
Mprime	6	**6**	**6**	4	4	4	**6**	4	**6**	4	0	28	4	1.0	1.0	2	0.8	0.5
Mystery	1	1	1	1	1	1	1	1	1	1	0	25	1	1.0	1.0	0		
NoMystery	20	14	14	**20**	18	18	14	**20**	**20**	14	0	1	14	1.0	1.0	8	0.9	0.2
Openstacks08	30	21	19	19	19	19	19	19	**29**	18	8	0	18	1.2	1.0	16	0.3	0.5
Openstacks11	20	16	14	14	14	14	14	14	**20**	13	9	0	13	1.2	1.0	13	0.3	0.4
Openstacks14	20	3	3	2	1	1	3	2	**12**	1	2	0	1	1.1	1.0	1	0.2	0.6
Organic-Split	16	**11**	**11**	9	7	7	**11**	7	5	7	0	13	7	1.0	1.0	7	0.9	0.6
ParcPrinter08	13	4	**13**	7	**13**	**13**	**13**	**13**	6	13	73	58	4	2183.4	24.4	3	1700.6	16.7
ParcPrinter11	10	3	**10**	6	**10**	**10**	**10**	**10**	4	10	74	62	3	2183.4	24.4	3	1717.1	16.7
Pathways	30	5	5	5	5	5	5	5	5	5	21	17	5	13.0	2.9	1	15.7	2.8
PSR	48	47	47	47	47	47	47	47	**48**	47	9	16	47	1.0	1.0	12	0.7	0.8
Rovers	40	8	9	8	10	10	9	10	**13**	9	26	28	8	257.6	48.7	4	180.6	39.3
Satellite	36	7	**12**	9	**12**	**12**	**12**	11	10	12	52	34	7	41.9	7.7	3	67.8	12.3
Scanalyzer08	9	5	5	5	4	4	5	5	**6**	4	0	13	4	1.0	1.0	1	0.7	0.1
Scanalyzer11	5	3	3	3	2	2	3	3	**4**	2	0	8	2	1.0	1.0	1	0.8	0.1
Tidybot11	20	**14**	**14**	**14**	13	13	**14**	**14**	**14**	13	18	8	13	1.5	1.0	13	1.3	0.3
Tidybot14	20	9	9	**10**	7	7	9	**10**	**10**	7	19	0	7	1.5	1.0	7	1.3	0.3
TPP	29	6	5	5	5	5	6	5	**14**	5	0	9	5	1.0	1.0	1	0.9	0.8
Transport08	12	9	9	9	9	9	9	9	9	9	2	5	9	1.0	1.0	5	0.9	0.8
Transport14	17	5	5	5	5	5	5	5	**7**	5	2	0	5	1.0	1.0	5	0.9	0.8
Woodworking08	26	14	**23**	16	20	20	**23**	20	19	20	60	45	13	31.8	25.2	6	469.4	1.2
Woodworking11	17	9	**16**	10	14	14	**16**	14	12	14	63	47	8	32.4	26.0	6	454.6	1.1
Zenotravel	20	**13**	**13**	12	12	12	**13**	12	**13**	12	2	11	12	1.1	1.0	6	1.0	1.0
Others	194	50	50	46	43	43	50	46	**85**	42	0	0	42	1.0	1.0	36	0.7	0.6
$\sum$	984	522	557	533	541	542	558	553	**583**									

Table 2: Detailed results for A* search with $h^{\text{LM-cut}}$ when using the incident-arcs factoring strategy. The setup is identical to Table 1. "C2" is Complementary 2 from IPC'18.

cluding ParcPrinter where due to the bad-performing baseline we have no improvement factors. This is even more than in standard search, where S^3 only improves runtime (coverage) in 5 (3) domains. However, it does not suffice to perform best in total coverage, where due to a less significant gain in ParcPrinter and Woodworking, and bad domains inherited from DS (mostly Elevators and Miconic), DS^3_O is 31 instances behind S^3 when using the safety belt.

Table 2 shows the results when instead of running blind search, we use A* search with the $h^{\text{LM-cut}}$ heuristic. One of the main differences is that here both S^3 and DS^3_O can actually improve over the respective baseline without using the safety belt, i.e., a smaller transition reduction already leads to a runtime improvement. Again, where S^3 improves coverage over B, we also see an improvement of DS^3_O over DS (ParcPrinter, Satellite, and Woodworking), with an exception in Driverlog. On the negative side, enabling the pruning in decoupled search cannot fully make up for the advantage in Woodworking (5 instances). This seems to be due to a rather bad problem decomposition of the incident-arcs factoring. When using a fork or inverted-fork factoring (Tables 3 and 4), DS^3 can solve all instances that S^3 solves.

	Coverage										Pruning Power											
								+safety b.			Transitions				Search Space				Runtime			
Domain	#	B	S^3	DS	DS^3	DS^3_O	~~DS^3_O~~	S^3	DS^3_O	C2	#	S^3	DS^3_O	~~DS^3_O~~	#	S^3	DS^3_O	~~DS^3_O~~	#	S^3	DS^3_O	~~DS^3_O~~
Driverlog	20	13	**14**	13	13	13	13	**14**	13	**14**	13	16	4	0	13	1.1	1.0	1.0	8	1.0	0.8	0.9
Logistics00	28	20	21	**28**	**28**	**28**	**28**	21	**28**	22	21	28	45	44	20	3.4	1.6	1.7	10	4.2	1.7	2.1
Logistics98	35	6	6	6	6	6	6	6	6	6	6	38	46	46	6	2.3	1.3	1.5	3	2.8	2.1	2.2
Maintenance	1	1	1	1	1	1	1	1	1	1	1	74	0	0	1	1.0	1.0	1.0	1	1.0	0.0	1.4
Miconic	145	**136**	**136**	135	135	135	135	**136**	135	99	135	0	3	0	135	1.0	1.0	1.0	91	0.8	0.9	1.0
NoMystery	20	14	14	**20**	18	18	19	14	**20**	**20**	14	0	1	0	14	1.0	1.0	1.0	8	0.9	0.1	0.3
Pathways	29	4	4	4	4	4	4	4	4	4	4	25	23	26	4	13.0	7.1	7.5	1	15.7	8.9	10.0
PSR	3	3	3	3	3	3	3	3	3	3	3	2	9	11	3	1.0	1.0	1.0	0			
Rovers	40	8	9	9	11	11	11	9	11	**13**	9	26	26	26	8	258	52.0	52.2	4	181	44.7	45.8
Satellite	36	7	**12**	7	**12**	**12**	11	**12**	**12**	10	11	51	47	34	7	41.9	26.4	7.0	3	67.8	40.5	11.2
TPP	27	6	5	**18**	**18**	**18**	**18**	6	**18**	14	5	0	23	21	5	1.0	1.0	1.0	1	0.9	1.0	1.0
Woodwor08	13	6	**11**	10	**11**	**11**	**11**	**11**	**11**	10	11	62	35	33	6	81.5	3.9	3.9	3	368	4.4	4.8
Woodwor11	5	2	**5**	4	**5**	**5**	**5**	**5**	**5**	4	5	65	50	47	2	82.8	3.9	3.9	2	370	4.8	4.8
Zenotravel	20	13	13	13	13	13	13	13	13	13	13	2	5	0	13	1.1	1.0	1.0	7	1.0	0.9	0.9
$\sum$	422	239	254	271	278	278	278	255	**280**	233												

Table 3: Detailed results for A* search with $h^{\text{LM-cut}}$ when using a fork factoring strategy. The setup is identical to Table 1.

Regarding the differences between DS^3 and DS^3_O, there are again almost none. The only instance solved by DS^3_O but not by DS^3 is solved in $1658s$, so again can be attributed to noise, in particular since (almost) no transitions are pruned. Comparing the two configurations in more detail reveals that the size of the search space of DS^3_O is only smaller than that of DS^3 in Logistics00. Here, the percentage of pruned transitions on commonly solved instances increases from 14% to 39%, the runtime improves by a factor of 2.

The pruning power is comparable to the blind search case. The percentage of transitions pruned is usually in the same range, but leads to a less pronounced reduction in the search space size for decoupled search. Nevertheless, this results in a speed-up of the search in many cases, because the relative overhead spent for computing the stubborn sets is smaller than when running blind search – it takes time to evaluate the $h^{\text{LM-cut}}$ heuristic. Thus, the overhead pays off even if the search space size reduction is only moderate.

With the safety belt, DS^3_O solves all instances solved by DS or DS^3_O, except two in Organic-Split, and one in Satellite. In Organic-Split, the safety belt does not trigger, because there are at most 1082 state expansions; the instance in Satellite is solved only barely within the time limit by the succeeding configurations.

To conclude this section, we want to analyze the special case optimizations from Section 5 in more detail. Table 3 sheds further light on the fork factorings. Here, we compare to the faulty implementation of Gnad et al. (2016), which does not guarantee completeness (~~DS^3_O~~). The main difference to our fork-optimized variant are the missing price-frontier sets for an action's leaf preconditions. Observe that the difference between the two versions is rather small. Due to the larger overhead, DS^3_O looses one instance in NoMystery which it gains back in Satellite. Other than that, the pruning power is similar. There are also a few cases where DS^3_O actually prunes slightly *more* transitions than ~~DS^3~~. This is due to a stronger pruning in goal states, where in DS^3_O transitions are not included that cannot possibly contribute to a lower goal cost of a leaf (Definition 9 (ii)). This is ignored in the special case definition of frontiers from Gnad et al. (2016), which corresponds to part (i) of the Definition.

	Coverage									Pruning Power								
							+safety b.			Transitions			Search Space			Runtime		
Domain	#	B	S^3	DS	DS^3	DS^3_O	S^3	DS^3_O	C2	#	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O
Depots	22	7	7	7	7	7	7	7	7	7	3	1	7	1.0	1.0	5	0.9	0.9
Elevators08	30	22	22	23	22	22	22	23	**25**	22	23	1	22	1.2	1.0	21	1.5	0.7
Elevators11	20	18	18	18	18	18	18	18	**19**	18	24	2	18	1.1	1.0	18	1.2	0.7
Logistics00	28	20	21	20	20	20	21	20	**22**	20	27	2	20	3.4	1.0	11	4.2	0.9
Logistics98	35	6	6	**7**	**7**	**7**	6	**7**	6	6	38	2	6	2.3	1.0	3	2.8	0.9
Maintenance	1	1	1	1	1	1	1	1	1	1	74	0	1	1.0	1.0	1	1.0	0.0
Rovers	38	6	7	5	7	7	7	7	**11**	7	33	26	5	3.4	5.2	3	2.6	3.1
Satellite	34	5	**10**	**10**	**10**	**10**	**10**	**10**	8	10	61	17	5	41.9	1.4	3	67.8	1.4
Transport08	30	11	11	11	11	11	11	11	**14**	11	2	4	11	1.0	1.0	7	0.9	0.9
Transport14	20	6	6	6	6	6	6	6	**9**	6	2	0	6	1.0	1.0	6	0.9	0.8
Woodworking08	24	12	**22**	18	**22**	**22**	**22**	**22**	17	21	62	40	12	302.7	20.7	6	530.3	26.0
Woodworking11	15	8	**15**	12	**15**	**15**	**15**	**15**	11	15	63	43	8	306.1	20.7	6	517.2	25.9
Zenotravel	18	11	11	11	11	11	11	11	11	11	3	5	11	1.1	1.0	7	1.0	0.6
Others	110	22	21	21	21	21	22	21	**55**	21	0	0	21	1.0	1.0	20	0.9	0.8
$\sum$	425	155	178	170	178	178	179	179	**216**									

Table 4: Detailed results for A* search with $h^{\text{LM-cut}}$ when using an inverted-fork factoring strategy. The setup is identical to Table 1.

When comparing DS^3 to DS^3_O, it turns out that, as before, there are mostly minor differences. In Logistics, the average percentage of pruned transitions increases from 16% for DS^3 to 40% with the optimizations on commonly solved instances. Other than that, the only difference is in Satellite, where the pruning increases by a bit, slightly reducing the size of the search space.

For inverted-fork factorings – Table 4 – we see a nice improvement when enabling the pruning in decoupled search, the coverage increases by 2 instances in Rovers, and by 8 in Woodworking. Coverage decreases only by one instance in Elevators. There is also a high reduction in search space size and runtime for Woodworking. Compared to S^3, DS^3_O cannot make up for one instance in Logistics00. Observe that the pruning power of decoupled strong stubborn sets is highly dependent on the factoring type. While with fork factorings, around 40% of the transitions in Logistics are pruned, there is almost no reduction when using an inverted-fork factoring. The same phenomenon is visible in Satellite. In both cases, the speed-up over DS either vanishes completely, or is significantly reduced.

Comparing DS^3 to DS^3_O, it turns out that there is absolutely no difference in terms of pruning power. Even the absolute number of pruned transitions does not change. There is a measurable speed-up in Satellite, though, where the average runtime improves by a factor of 1.9, due to a more efficient computation of the stubborn sets. Here, the special-case optimization for single-variable inverted-fork leaves, which is described in the Appendix, triggers. Other than that, the special case optimizations apparently do not lead to an increased pruning power in the domains we are using.

In summary, we have seen similar results throughout this section. In many cases, though not in all domains, we get the desired improvement of DS^3/DS^3_O over DS in domains when S^3 improves over B. The safety belt mechanism is a nice and simple way to get the best of both worlds, i.e., when stubborn sets pruning is effective, it can be used, if not, we can easily disable the pruning to prevent its computational overhead. Unfortunately, it looks like the special case handling for fork and inverted-fork leaves can only add little over the base definitions from Section 4 on our benchmark set.

	Coverage								Pruning Power								
							+safety b.		Transitions			Search Space			Runtime		
Domain	#	B	S^3	DS	DS^3	DS^3_O	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O
Unsolvability IPC'16 domains																	
Cavediving	21	3	3	**4**	**4**	**4**	3	**4**	2	11	4	2	1.1	1.0	2	0.1	0.8
Diagnosis	11	4	5	5	**7**	**7**	5	**7**	5	52	47	4	7.7	11.6	3	2.6	6.4
DocumentTransfer	20	5	5	5	5	5	5	5	5	7	14	5	2.3	1.4	5	0.3	1.0
NoMystery	24	2	2	**12**	11	11	2	**12**	2	48	0	2	4.3	1.0	2	0.5	0.0
Rovers	20	7	6	**9**	7	7	6	7	6	39	25	6	1.7	1.5	6	0.2	0.2
TPP	30	**17**	14	11	9	9	**17**	11	9	16	3	9	1.0	1.0	9	0.1	0.2
PegSol-R5	12	2	2	2	2	2	2	2	2	24	23	2	1.0	1.0	1	0.3	0.4
PegSol	24	24	24	24	24	24	24	24	24	1	1	24	1.0	1.0	14	0.3	0.7
Others	46	**22**	13	17	9	9	**22**	17	9	0	0	9	1.0	1.0	9	0.1	0.0
Unsolvable benchmarks from Hoffmann, Kissmann, and Torralba (2014)																	
3SAT	1	1	1	1	1	1	1	1	1	78	76	1	106.0	69.1	1	103.6	232.6
Others	75	6	1	**23**	15	15	3	**23**	0			0			0		
$\sum$	284	93	76	**113**	94	94	90	**113**									

Table 5: Detailed results for blind search when using the incident-arcs factoring strategy. The setup is identical to Table 1. Here, "coverage" stands for the number of instances proved unsolvable.

Compared to Complementary 2, as a representative of the state-of-the-art in optimal planning, there are a few cases where DS^3_O performs better. This is the case in Logistics and ParcPrinter, when using the incident-arcs factorings; additionally in Satellite, TPP, and Woodworking for fork factorings, and in Satellite with inverted-fork factorings. We conclude that decoupled strong stubborn sets can compete with state-of-the-art optimal planners when a good problem decomposition is possible, and stubborn-sets pruning is effective.

7.3 Proving Unsolvability

We will next illustrate the potential of decoupled strong stubborn sets for proving unsolvability of planning tasks. We use two different configurations. Table 5 shows results for blind search with the incident-arcs factoring, where the entire state space is exhausted, proving the absence of goal states reachable from the initial state. In Table 6, we additionally use the h^{max} heuristic as a dead-end detector (Bonet & Geffner, 2001). For all planners based on decoupled search we switch to a pricing function that only distinguishes between reachable (price 0) and unreachable leaf states (price ∞). We show results on all domains from the Unsolvability IPC'16 and those from Hoffmann et al. (2014) where not all instances have been reused for the competition. When using h^{max}, we also compare against Sympa, the best non-portfolio planner in the Unsolvability IPC'16 (Torralba, 2016).

Table 5 illustrates that in the domains at hand, although we see some significant transition pruning that translates into a reduction of the search space in a couple of domains, this only leads to a runtime improvement in Diagnosis and 3SAT. Coverage increases in Diagnosis, and we get a quite remarkable result in 3SAT, where runtime decreases from around $260s$ for DS to only $1.1s$ for DS^3_O. Unfortunately, only a single instance of this domain can be decomposed using the incident-arcs strategy. Again, we do not see a difference between DS^3 and DS^3_O in terms of coverage, nor in the number of transitions pruned, or runtime. The safety belt is able to recover all but two instances in Rovers, when using decoupled search. For standard search, it fails in 4 instances.

	Coverage									Pruning Power								
							+safety b.			Transitions			Search Space			Runtime		
Domain	#	B	S^3	DS	DS^3	DS^3_O	S^3	DS^3_O	Sympa	#	S^3	DS^3_O	#	S^3	DS^3_O	#	S^3	DS^3_O
Unsolvability IPC'16 domains																		
Cavediving	21	3	3	**4**	**4**	**4**	3	**4**	3	2	5	1	2	1.1	1.0	2	0.3	0.9
Diagnosis	11	5	8	8	**10**	**10**	8	**10**	9	8	43	27	5	1.1	1.0	1	0.7	0.6
Rovers	20	7	7	10	9	9	7	10	**18**	7	2	4	7	1.3	1.3	7	0.2	0.2
PegSol	24	24	24	24	24	24	24	24	24	24	1	0	24	1.0	1.0	10	0.5	0.8
Others	132	40	33	54	49	49	40	53	**62**	32	0	0	32	1.0	1.0	27	0.3	0.3
Unsolvable benchmarks from Hoffmann et al. (2014)																		
3SAT	1	1	1	1	1	1	1	1	0	1	77	74	1	117.4	82.2	1	167.6	188.0
Rovers	25	2	1	2	2	2	2	2	**19**	1	1	1	1	1.1	1.2	1	0.2	0.6
Others	50	6	2	24	20	20	6	24	**40**	0			0			0		
$\sum$	284	88	79	127	119	119	91	128	**175**									

Table 6: Detailed results for A* search with h^{max} when using an incident-arcs factoring strategy. The setup is identical to Table 1. Here, "coverage" stands for the number of instances proved unsolvable.

The additional usage of h^{max} as a dead-end detector does not change the big picture. In Diagnosis and 3SAT, the stubborn sets pruning can improve the results of DS significantly. In Diagnosis, this is a bit hidden, because of only little pruning happening in the single (non-trivially) commonly solved instance. Comparing DS and DS^3_O directly we observe a runtime improvement factor of 16, and two more instances solved. Activating the safety belt in this configuration results in the best of DS and DS^3_O for decoupled search in all but one instance.

DS^3_O can compete with Sympa in Cavediving, Diagnosis, and the resource-constrained variant of NoMystery. The big advantage of Sympa in terms of total coverage stems from the resource-constrained Rovers and TPP domains. NoMystery and TPP are not shown in the tables, since no pruning happens in these domains. Both appear in the UIPC'16 benchmarks as well as the domains from Hoffmann et al. (2014), though with different instances. Across both benchmark sets, DS^3_O (with safety belt) proves 13 instances unsolvable, Sympa 38, in TPP. In NoMystery, it is 36 instances for DS^3_O, and 35 for Sympa.

In summary, strong stubborn sets pruning can be highly beneficial for proving the unsolvability of planning tasks with decoupled search, if the tasks have the required structure. In the domains that we use for evaluation, this does not happen frequently, though. Like in optimal planning, when using a safety belt mechanism, we can nicely combine the two techniques and inherit the strengths of both components – strong stubborn sets and decoupled search.

8. Conclusion

Partial-order reduction via strong stubborn sets is a well-established technique that has originally been introduced in model checking and recently caught interest in the planning community. It is designed to avoid the exploration of redundant parts of the state space that result from the application of different permutations of action sequences. Star-topology decoupled search is a novel approach to exploit conditionally independent parts of a planning problem by decomposition. Both techniques have been proposed to tackle the state space explosion problem inherent to many variants of explicit

state space search. In this work, we combine the two methods in the context of classical planning to be able to get the best of both worlds.

We extend our previous work that has developed strong stubborn sets pruning in the restricted setting when using fork factorings, fixing a special case handling that can lead to unsafe pruning.[7] Moving from fork to general star topologies is a highly non-trivial extension that needs a careful analysis of the possible interaction between center and leaf factors. As such, it also sacrifices some pruning potential, not only from this more complex dependencies, but also given that the pruning is restricted to center actions.

Nevertheless, our experimental evaluation illustrates that enabling strong stubborn sets pruning in decoupled search can be highly beneficial, in both optimal planning and proving unsolvability. By using a simple safety belt mechanism, we are able to switch off the pruning on instances where it is not useful. This skips the expensive computation of decoupled strong stubborn sets, leading to a superior overall performance.

With the complete integration of strong stubborn sets pruning into decoupled search, the question arises if we can combine this with other techniques. Recently, symmetry breaking (Gnad et al., 2017c) and symbolic representations (Gnad, Torralba, & Hoffmann, 2017b) have been adapted to the decoupled setting. Can those be combined with decoupled strong stubborn sets?

And finally, having adopted partial-order reduction from model checking, why not apply decoupled strong stubborn sets to this problem? In directed model checking, search algorithms suffer from the same exponential blow-up of the search space like planning. And since it has already been shown that star-topology decoupled search can be applied successfully in directed model checking (Gnad, Dubbert, Lluch-Lafuente, & Hoffmann, 2018), we can further enhance it by the combination we propose in this work.

Acknowledgments. Daniel Gnad was supported by the German Research Foundation (DFG), under grant HO 2169/6-1, "Star-Topology Decoupled State Space Search". Jörg Hoffmann's research group has received support by DFG grant 389792660 as part of TRR 248 (see `https://perspicuous-computing.science`).

Appendix A. Proofs

A.1 DSSS Special Case Topologies

Lemma 4. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^{\mathcal{F}}$ be a solvable decoupled state, let F^L be a leaf factor, and let p be a partial assignment to F^L reached in $s^{\mathcal{F}}$. Let $\pi^{\mathcal{F}}$ be a decoupled plan for $s^{\mathcal{F}}$, and let π be a global plan given $\pi^{\mathcal{F}}$. Let A be a fork-price frontier set for p in $s^{\mathcal{F}}$.*

Define a_t, π^{past}, and π^{future} as before. If there exists $k > t$ such that, denoting by $\langle a_1^L, \dots, a_i^L \rangle$ and $s_0^L, \dots, s_i^L$ the F^L actions, respectively states, in π prior to a_k, we have $cost(\langle a_1^L, \dots, a_i^L \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s_i^L)$ and $s_i^L \models p$, then $\{a_t, \dots, a_{k-1}\} \cap A \neq \emptyset$.

Proof. Denote by l the index of the first action on π^L where a_l^L is on π^{future}. Since we have that $cost(\langle a_1^L, \dots, a_i^L \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s_i^L)$ by prerequisite, a_i^L must be on π^{future}, i.e., we must have $i \geq l$. Let $j \geq l$ be the smallest index where $cost(\langle a_1^L, \dots, a_j^L \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s_j^L)$. Consider

7. We remark that, although possible in theory, this never lead to suboptimal solutions or instances falsely identified as unsolvable in our experiments

the transition $s^L_{j-1} \xrightarrow{a^L_j} s^L_j$ on π^L. As $j \geq l$, this transition is part of π^{future}. We prove that *(*) $s^L_{j-1} \xrightarrow{a^L_j} s^L_j$ is a fork-price frontier transition for p in $s^{\mathcal{F}}$.*

Because j is the smallest index where $cost(\langle a^L_1, \ldots, a^L_j \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s^L_j)$, it follows that $\mathsf{prices}[s^{\mathcal{F}}](s^L_{j-1}) \leq cost(\langle a^L_1, \ldots, a^L_{j-1} \rangle)$. Observe that, hence, s^L_{j-1} is reached in $s^{\mathcal{F}}$, as its price is upper-bounded by a finite value. Further, observe that, adding the cost of a^L_j on both sides of the inequality $\mathsf{prices}[s^{\mathcal{F}}](s^L_{j-1}) \leq cost(\langle a^L_1, \ldots, a^L_{j-1} \rangle)$, we get $\mathsf{prices}[s^{\mathcal{F}}](s^L_{j-1}) + cost(a^L_j) \leq cost(\langle a^L_1, \ldots, a^L_j \rangle)$. As, by construction, $cost(\langle a^L_1, \ldots, a^L_j \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s^L_j)$, we obtain that $\mathsf{prices}[s^{\mathcal{F}}](s^L_{j-1}) + cost(a^L_j) < \mathsf{prices}[s^{\mathcal{F}}](s^L_j)$.

Furthermore, $s^L_{j-1} \xrightarrow{a^L_j} s^L_j$ lies on the F^L path π^L from $s_0[F^L]$ to s^L_i where $s^L_i \models p$. To show (*), it remains to prove that (a) $\mathsf{center}[s^{\mathcal{F}}] \not\models pre(a^L_j)[F^C]$ and (b) π^L is simple.

Regarding (a), because a^L_j affects F^L, in a fork topology a^L_j cannot affect the center. Hence, $\mathsf{prices}[s^{\mathcal{F}}](s^L_{j-1}) + cost(a^L_j) < \mathsf{prices}[s^{\mathcal{F}}](s^L_j)$ implies that a^L_j cannot be reached in $s^{\mathcal{F}}$. We do know, however, that s^L_{j-1} is reached in $s^{\mathcal{F}}$. As the only non-F^L precondition of a^L_j must be on F^C, (a) follows.

Regarding (b), in a fork topology, any cheapest compliant leaf path is simple, because without center preconditions nor effect on the leaf there is no reason to visit the same leaf state twice.

We have now proved (*). Given this, as A is a goal-price frontier set for F^L in $s^{\mathcal{F}}$, by definition we know that the segment $\langle a^L_j, \ldots, a^L_i \rangle$ of π^L between s^L_{j-1} and s^L_i must contain an action from A. The claim follows as, with $j \geq l$ and by definition of a^L_i, $\langle a^L_j, \ldots, a^L_i \rangle$ is a subsequence of $\langle a_t, \ldots, a_{k-1} \rangle$. □

Definition 13 (Fork-Past-Maximality)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^{\mathcal{F}}$ be a decoupled state, let $\pi^{\mathcal{F}}$ be a decoupled plan for $s^{\mathcal{F}}$, and let π be a global plan given $\pi^{\mathcal{F}}$.*

*Define a_t, π^{past}, and π^{future} as before. We say that π is **fork-past-maximal** if, for every leaf factor F^L and for every $k > t$ where $a_k \in \mathcal{A}^L[F^L]$ is reached in $s^{\mathcal{F}}$, denoting by $\langle a^L_1, \ldots, a^L_i \rangle$ and $s^L_0, \ldots, s^L_i$ the F^L actions, respectively states, in π prior to a_k, we have $cost(\langle a^L_1, \ldots, a^L_i \rangle) < \mathsf{prices}[s^{\mathcal{F}}](s^L_i)$.*

Lemma 5. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^{\mathcal{F}}$ be a decoupled state, let $\pi^{\mathcal{F}}$ be a decoupled plan for $s^{\mathcal{F}}$, and let π be a global plan given $\pi^{\mathcal{F}}$. Then there exists a fork-past-maximal global plan π' given $\pi^{\mathcal{F}}$ where $cost(\pi') \leq cost(\pi)$.*

Proof. We obtain such a π' as follows. Start with $\pi' := \pi$. If π' is past-maximizing, stop. Else, select a counter-example F^L and k, and denote $\langle a^L_1, \ldots, a^L_i \rangle$ and $s^L_0, \ldots, s^L_i$ as in Definition 13. As $cost(\langle a^L_1, \ldots, a^L_i \rangle) \geq \mathsf{prices}[s^{\mathcal{F}}](s^L_i)$, there exists a $\pi^C[s^{\mathcal{F}}]$-compliant F^L path π^L_i that ends in s^L_i. So π' remains a plan when removing $\langle a^L_1, \ldots, a^L_i \rangle$ from π', and inserting π^L_i as a subsequence of π^{past}. Further, we can move a_k to the end of π^{past}, because its F^L precondition is achieved by π^L, and its F^C precondition (if any) is true in $s_0[\![\pi^{past}]\!]$ as $pre(a_k)$ is reached in $s^{\mathcal{F}}$. Now, iterate. This algorithm terminates as, after each step, there is one action less in π^{future}. The outcome π' satisfies the claim as we have not changed the center-action subsequence, and the permuted leaf paths are still compliant with that sequence. □

Theorem 3. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a fork factoring. Let $s^\mathcal{F}$ be a solvable decoupled state for which $\langle\rangle$ is not an optimal decoupled plan, and let $\pi^\mathcal{F}$ be a strongly optimal decoupled plan for $s^\mathcal{F}$. Let $\mathcal{A}_s^\mathcal{F}$ be an FDSSS for $s^\mathcal{F}$.*

Then $\mathcal{A}_s^\mathcal{F}$ contains a center action starting a permutation of $\pi^\mathcal{F}$.

Proof. Let π be a global plan for $\pi^\mathcal{F}$, and assume, without loss of generality by Lemma 5, that π is fork-past-maximal. Denote $\pi = \langle a_1, \dots, a_n \rangle$. As above, let a_t be the starting action of $\pi^C[\pi^\mathcal{F}]$ in π, and denote $\pi^{past} := \langle a_1, \dots, a_{t-1} \rangle$ and $\pi^{future} := \langle a_t, \dots, a_n \rangle$. Then the following properties hold:

(a) There must be an action a shared between $\mathcal{A}_s^\mathcal{F}$ and π^{future}, i.e., $a \in \mathcal{A}_s^\mathcal{F} \cap \{a_t, \dots, a_n\}$.

If $s_\star$ is not reached in $s^\mathcal{F}$, then this follows by the same argument as for (a) in the proof of Theorem 1.

If $s_\star$ is reached in $s^\mathcal{F}$, then this follows by the same argument as for (a) in the proof of Theorem 2, replacing Lemma 3 with the variant of Lemma 4 for fork-goal-price frontier sets.

Given (a), let a_k be the first shared action, i.e., say that $a_k \in \mathcal{A}_s^\mathcal{F}$ and $\{a_t, \dots, a_{k-1}\} \cap \mathcal{A}_s^\mathcal{F} = \emptyset$.

(b) a_k is reached in $s^\mathcal{F}$.

By the same argument as for (b) in the proof of Theorem 1.

(c) a_k is a center action, $a_k \in \mathcal{A}^C$.

Assume for contradiction that a_k is a leaf action, $a_k \in \mathcal{A}^L[F^L]$. First, with (b) and due to Definition 10, we then know that $\mathcal{A}_s^\mathcal{F}$ contains a fork-price frontier set A for $p := pre(a_k)[F^L]$ in $s^\mathcal{F}$. We show that π^{future} contains an action from A in front of a_k, in contradiction to a_k being the first shared action.

Denote by $\langle a_1^L, \dots, a_i^L \rangle$ and $s_0^L, \dots, s_i^L$ the F^L actions, respectively states, in π prior to a_k. We know that $k \geq t$ because a_k is on π^{future}. In fact, as a_t is a center action, given the fork topology we know that $a_k \neq a_t$ and hence $k > t$. Given this, as π is fork-past-maximal and a_k is reached in $s^\mathcal{F}$ by (b), we have that $cost(\langle a_1^L, \dots, a_i^L \rangle) < \mathsf{prices}[s^\mathcal{F}](s_i^L)$. Furthermore, of course $s_i^L \models p = pre(a_k)[F^L]$. We can therefore apply Lemma 4 and get that $\{a_t, \dots, a_{k-1}\} \cap A \neq \emptyset$, as we needed to show.

(d) a_k does not interfere with any of the actions a_i, $t \leq i \leq k-1$, where $pre(a) \parallel pre(a_i)$.

With (b) and (c), and as Definition 10 (iii) includes interfering actions for reached center actions in $\mathcal{A}_s^\mathcal{F}$, this follows by the same argument as for (c) in the proof of Theorem 1.

(e) a_k can be moved to the start of π^{future}. Precisely, $\pi' := \pi^{past} \circ \langle a_k, a_t, \dots, a_{k-1}, a_{k+1}, \dots, a_n \rangle$ is a plan for Π.

By the same argument as for (e) in the proof of Theorem 1, except that we do not need to take care of leaf preconditions as, in a fork structure, the center action a_k cannot have any such preconditions.

The claim now follows with the same concluding argument as in the proof of Theorem 1. □

Example 3. *We give an example $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$, fork factoring $\mathcal{F}$, and solvable non-goal $s^\mathcal{F}$, where an FDSSS $\mathcal{A}_s^\mathcal{F}$ as per Definition 10 but without item (iv) does not contain the starting action of any decoupled plan for $s^\mathcal{F}$.*

We define Π as follows:

- $\mathcal{V} = \{l_1, l_2, c_1, c_2\}$, *with* $\mathcal{D}(l_1) = \{0, 1, 2, 3\}$ *and* $\mathcal{D}(l_2) = \mathcal{D}(c_1) = \mathcal{D}(c_2) = \{0, 1\}$.
- $s_0(v) = 0$ *for all* $v \in \mathcal{V}$.
- $s_\star = \{l_1 = 3, l_2 = 0\}$.
- *The action set $\mathcal{A}$ contains:*

  $a^{C_1}_{0\to1}$: *precondition* $\{c_1 = 0\}$ *effect* $\{c_1 = 1\}$
  $a^{C_2}_{0\to1}$: *precondition* $\{c_2 = 0\}$ *effect* $\{c_2 = 1\}$
  $a^{L}_{0\to1}$: *precondition* $\{l_1 = 0, c_1 = 1\}$ *effect* $\{l_1 = 1\}$
  $a^{L}_{1\to2}$: *precondition* $\{l_1 = 1, c_2 = 0\}$ *effect* $\{l_1 = 2\}$
  $a^{L}_{2\to3}$: *precondition* $\{l_1 = 2, c_2 = 1\}$ *effect* $\{l_1 = 3\}$
  $a^{L\bot}_{0\to1}$: *precondition* $\{l_1 = 0\}$ *effect* $\{l_1 = 1, l_2 = 1\}$

The factoring $\mathcal{F}$ has the center $F^C = \{c_1, c_2\}$, and the single leaf $F^L = \{l_1, l_2\}$. Figure 2 illustrates the state spaces of these factors.

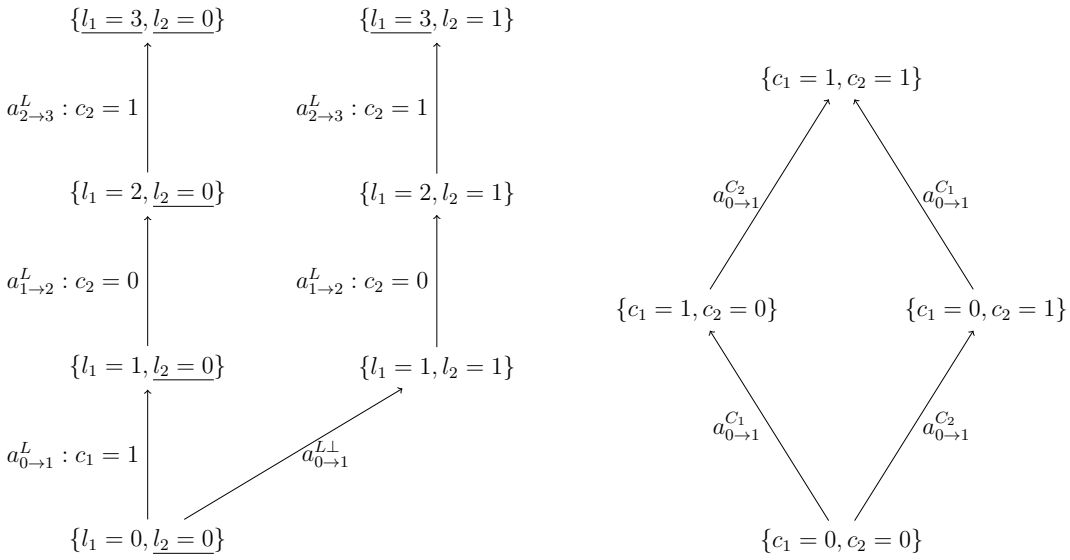

Figure 2: The leaf (left) and center (right) state spaces in the planning task from Example 3. Underlined variable values are goal values; all variables initially have value 0.

The only way to solve this task is the leaf path $\langle a^L_{0\to1}, a^L_{1\to2}, a^L_{2\to3} \rangle$. To execute that leaf path, we must first apply the center action $a^{C_1}_{0\to1}$, achieving the precondition of $a^L_{0\to1}$; then we need to apply $a^L_{0\to1}$, as well as $a^L_{1\to2}$ which has precondition $c_2 = 0$; and only then can we apply $a^{C_2}_{0\to1}$, achieving the precondition of the final action $a^L_{2\to3}$. Note that we can not *apply the center actions*

in the opposite order: we must first move c_1, because once we moved c_2 to 1 we cannot move it back to 0.

We set $s^{\mathcal{F}}$ to the initial decoupled state here, $s^{\mathcal{F}} := s_0^{\mathcal{F}}$. Due to the leaf action $a_{0\to1}^{L\perp}$, $l_1 = 1$ is reached in $s^{\mathcal{F}}$; the only unreached leaf-variable values are $l_1 = 2$ and $l_2 = 3$. Given the above, the only decoupled plan for $s^{\mathcal{F}}$ is induced by the center-action sequence $\langle a_{0\to1}^{C_1}, a_{0\to1}^{C_2}\rangle$.

So, what does the construction of an FDSSS $\mathcal{A}_s^{\mathcal{F}}$ do here? Say first that we use Definition 10 without item (iv). By Definition 10 (i), we first collect all actions achieving $l_1 = 3$ into $\mathcal{A}_s^{\mathcal{F}}$. There is only one such action, namely $a_{2\to3}^L$. For that action, the leaf precondition $l_1 = 2$ is not reached, so the necessary enabling set as per Definition 10 (ii) adds the only achiever, $a_{1\to2}^L$, into $\mathcal{A}_s^{\mathcal{F}}$. That action is reached – its leaf precondition $l_1 = 1$ is reached in $s^{\mathcal{F}}$ – so without Definition 10 (iv) it spawns no further activity. The construction stops with $\mathcal{A}_s^{\mathcal{F}} = \{a_{2\to3}^L, a_{1\to2}^L\}$. As advertised, this does not contain the starting action of any decoupled plan for $s^{\mathcal{F}}$ – indeed it does not contain any center action at all.

The issue here is that, with $pre(a_{1\to2}^L) = \{l_1 = 1\}$ already being reached in $s^{\mathcal{F}}$, without item (iv) we overlook that, yes, this condition is reached, but in the wrong leaf state, namely $\{l_1 = 1, l_2 = 1\}$ which is not part of any solution.

For the record, if we do use item (iv) of Definition 10, then this issue is fixed: with $p := \{l_1 = 1\}$, the leaf state $r^L := \{l_1 = 1, l_2 = 0\}$, i.e., the leaf state we actually need to use on a solution, forces a fork-price frontier set for p in $s^{\mathcal{F}}$ to contain $a_{0\to1}^L$. To see this, observe that $r^L \models p$; that the path $\pi^L = \langle a_{0\to1}^L\rangle$ from $s_0[F^L]$ to $\{l_1 = 1, l_2 = 0\}$ has $cost(\pi^L) = 1 < \infty = \mathsf{prices}[s^{\mathcal{F}}](r^L)$; and that by setting $s^L := s_0[F^L]$ and $t^L := r^L$ the transition $s^L \xrightarrow{a_{0\to1}^L} t^L$ is a fork-price frontier transition for p in $s^{\mathcal{F}}$.

Definition 14 (DSSS: Enhanced)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star\rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Denote by $\mathcal{F}^{FL}$ the set of fork leaves in $\mathcal{F}$, denote by $\mathcal{F}^{IFL}$ the set of inverted-fork leaves in $\mathcal{F}$, and denote $\mathcal{F}^{GL} := \mathcal{F}^L \setminus (\mathcal{F}^{FL} \cup \mathcal{F}^{IFL})$. Let $s^{\mathcal{F}}$ be a decoupled state where $s_\star[\bigcup_{F^L\in\mathcal{F}^{IFL}} F^L]$ is reached in $s^{\mathcal{F}}$.*

*An action set $\mathcal{A}_s^{\mathcal{F}}$ is an **enhanced decoupled strong stubborn set (EDSSS)** for $s^{\mathcal{F}}$ if all of the following conditions hold:*

- *(i) If $s_\star$ is not reached in $s^{\mathcal{F}}$, then $\mathcal{A}_s^{\mathcal{F}}$ contains a decoupled necessary enabling set for $s_\star$ in $s^{\mathcal{F}}$, discarding actions a' where $pre(a')[\bigcup_{F^L\in\mathcal{F}^{IFL}} F^L]$ is not reached in $s^{\mathcal{F}}$.*

  If $s_\star$ is reached in $s^{\mathcal{F}}$, then for every $F^L \in \mathcal{F}^{GL}$ where $s_\star[F^L] \neq \emptyset$, $\mathcal{A}_s^{\mathcal{F}}$ contains a goal-price frontier set for F^L in $s_\star^{\mathcal{F}}$, discarding actions a' where $pre(a')[\bigcup_{F^L\in\mathcal{F}^{IFL}} F^L]$ is not reached in $s^{\mathcal{F}}$; and for every $F^L \in \mathcal{F}^{FL}$ where $s_\star[F^L] \neq \emptyset$, $\mathcal{A}_s^{\mathcal{F}}$ contains a fork-goal-price frontier set for F^L in $s_\star^{\mathcal{F}}$.

- *(ii) For all actions $a \in \mathcal{A}_s^{\mathcal{F}}$ not reached in $s^{\mathcal{F}}$, $\mathcal{A}_s^{\mathcal{F}}$ contains a decoupled necessary enabling set for $pre(a)$ in $s^{\mathcal{F}}$, discarding actions a' where $pre(a')[\bigcup_{F^L\in\mathcal{F}^{IFL}} F^L]$ is not reached in $s^{\mathcal{F}}$.*

- *(iii) For all actions $a \in \mathcal{A}_s^{\mathcal{F}} \setminus \bigcup_{F^L\in\mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ reached in $s^{\mathcal{F}}$, $\mathcal{A}_s^{\mathcal{F}}$ contains all actions a' interfering with a where $pre(a) \parallel pre(a')$ and $pre(a')[\bigcup_{F^L\in\mathcal{F}^{IFL}} F^L]$ is reached in $s^{\mathcal{F}}$.*

- *(iv) For all actions $a \in \mathcal{A}_s^{\mathcal{F}} \setminus \bigcup_{F^L\in\mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ reached in $s^{\mathcal{F}}$, and for all F^L where $pre(a)[F^L] \neq \emptyset$, $\mathcal{A}_s^{\mathcal{F}}$ contains a reached-enabling set for $pre(a)[F^L]$ in $s^{\mathcal{F}}$; and for all actions $a^L \in \mathcal{A}_s^{\mathcal{F}} \cap \bigcup_{F^L\in\mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ reached in $s^{\mathcal{F}}$, $\mathcal{A}_s^{\mathcal{F}}$ contains a fork-price frontier set for $pre(a^L)[F^L]$ in $s^{\mathcal{F}}$*

Note that, in (i), actions in a fork-goal-price frontier set cannot have a precondition on an inverted-fork leaf, so we do not need to exclude actions with unreached inverted-fork leaf preconditions. In (iv), for non-fork-leaf actions a (first case in the item), a cannot have a precondition on any such leaf. Reached-enabling sets are thus collected only for non-fork leaves.

Theorem 6. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ a star factoring. Denote $\mathcal{F}^{FL}$, $\mathcal{F}^{IFL}$, and $\mathcal{F}^{GL}$ as in Definition 14. Let $s^{\mathcal{F}}$ be a solvable decoupled state (in particular, $s_\star[\bigcup_{F^L \in \mathcal{F}^{IFL}} F^L]$ is reached in $s^{\mathcal{F}}$), and let $\pi^{\mathcal{F}}$ be a decoupled plan for $s^{\mathcal{F}}$. Let $\mathcal{A}_s^{\mathcal{F}}$ be an EDSSS for $s^{\mathcal{F}}$.*

Then $\mathcal{A}_s^{\mathcal{F}}$ contains a center action starting a permutation of $\pi^{\mathcal{F}}$.

Proof. Let π be a global plan for $\pi^{\mathcal{F}}$. Observe first that the notions of past-maximality and fork-past-maximality affect individual leaves only, i.e., the actions a_k in question affect exactly one leaf factor. So we can transform π as per Lemma 1 to be past-maximal for non-fork leaves; and we can transform it as per Lemma 5 to be fork-past-maximal for fork leaves. We can hence assume that π has these properties, without loss of generality.

Denote $\pi = \langle a_1, \ldots, a_n \rangle$. As before, let a_t be the starting action of $\pi^C[\pi^{\mathcal{F}}]$ in π, and denote $\pi^{past} := \langle a_1, \ldots, a_{t-1} \rangle$ and $\pi^{future} := \langle a_t, \ldots, a_n \rangle$. Then the following properties hold:

(a) There must be an action a shared between $\mathcal{A}_s^{\mathcal{F}}$ and π^{future}, i.e., $a \in \mathcal{A}_s^{\mathcal{F}} \cap \{a_t, \ldots, a_n\}$.

If $s_\star$ is not reached in $s^{\mathcal{F}}$, this holds by the same argument as for (a) in the proof of Theorem 1.

If $s_\star$ is reached in $s^{\mathcal{F}}$, then for non-fork leaves this holds by the same argument as for (a) in the proof of Theorem 2, and for fork leaves this holds by the same argument as for (a) in the proof of Theorem 3.

In all but the last case, we invoke Proposition 1 to show that actions with unreached inverted-leaf preconditions can be discarded.

Given (a), let a_k be the first shared action, i.e., say that $a_k \in \mathcal{A}_s^{\mathcal{F}}$ and $\{a_t, \ldots, a_{k-1}\} \cap \mathcal{A}_s^{\mathcal{F}} = \emptyset$.

(b) a_k is reached in $s^{\mathcal{F}}$.

By the same argument as for (b) in the proof of Theorem 1, invoking Proposition 1 to show that actions with unreached inverted-leaf preconditions can be discarded.

(c) If $a_k \in \mathcal{A} \setminus \bigcup_{F^L \in \mathcal{F}^{FL}} \mathcal{A}^L[F^L]$, then a_k does not interfere with any of the actions a_i, $t \leq i \leq k-1$, where $pre(a) \parallel pre(a_i)$.

By the same argument as for (c) in the proof of Theorem 1, applied to $a_k \in \mathcal{A} \setminus \bigcup_{F^L \in \mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ with Definition 14 (iii), and invoking Proposition 1 to show that actions with unreached inverted-leaf preconditions can be discarded.

(d) $a_k \in \mathcal{A} \setminus \bigcup_{F^L \in \mathcal{F}^{FL}} \mathcal{A}^L[F^L]$.

Assume for contradiction that a_k is a fork-leaf action, $a_k \in \mathcal{A}^L[F^L]$ for a fork leaf F^L. With (b) and due to Definition 14, $\mathcal{A}_s^{\mathcal{F}}$ contains a fork-price frontier set A for $p := pre(a_k)[F^L]$ in $s^{\mathcal{F}}$. From here, the argument is the same as for (c) in the proof of Theorem 3, invoking fork-past-maximality for F^L.

(e) The leaf precondition of a_k is true at the end of π^{past}, i.e., in $s_0[\![\pi^{past}]\!]$.

By the same argument as for (d) in the proof of Theorem 1, applied to $a_k \in \mathcal{A} \setminus \bigcup_{F^L \in \mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ with (d) and Definition 14 (iv).

(f) a_k can be moved to the start of π^{future}. Precisely, $\pi' := \pi^{past} \circ \langle a_k, a_t, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n \rangle$ is a plan for Π.

By the same argument as for (e) in the proof of Theorem 1, applied to $a_k \in \mathcal{A} \setminus \bigcup_{F^L \in \mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ with (d).

(g) a_k is a center action.

By the same argument as for (e) in the proof of Theorem 1, applied to $a_k \in \mathcal{A} \setminus \bigcup_{F^L \in \mathcal{F}^{FL}} \mathcal{A}^L[F^L]$ with (d), invoking past-maximality for the non-fork leaf F^L.

The claim now follows with the same concluding argument as in the proof of Theorem 1. □

Definition 15 (DSSS: Inverted Forks with Single Variables)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ an inverted-fork factoring where every leaf consists of a single variable, $\forall F^L \in \mathcal{F}^L : |F^L| = 1$. Let $s^\mathcal{F}$ be a non-goal decoupled state where $s_\star[\mathcal{V} \setminus F^C]$ is reached in $s^\mathcal{F}$.*

*An action set $\mathcal{A}_s^\mathcal{F}$ is a **single-variable-inverted-fork decoupled strong stubborn set (SVIFDSSS)** for $s^\mathcal{F}$ if all of the following conditions hold:*

(i) $\mathcal{A}_s^\mathcal{F}$ contains a center-necessary enabling set for $s_\star[F^C]$ in $s^\mathcal{F}$.

(ii) For all center actions $a \in \mathcal{A}_s^\mathcal{F}$ not reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains a center-necessary enabling set for $pre(a)[F^C]$ in $s^\mathcal{F}$.

(iii) For all center actions $a \in \mathcal{A}_s^\mathcal{F}$ reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains all center actions a' interfering with a where $pre(a) \parallel pre(a')$ and $pre(a')[\mathcal{V} \setminus F^C]$ is reached in $s^\mathcal{F}$.

(iv) For all center actions $a \in \mathcal{A}_s^\mathcal{F}$ reached in $s^\mathcal{F}$, $\mathcal{A}_s^\mathcal{F}$ contains all center actions a' that have a leaf precondition competing with a, $pre(a)[\mathcal{V} \setminus F^C] \nparallel pre(a')[\mathcal{V} \setminus F^C]$, and $pre(a')[\mathcal{V} \setminus F^C]$ is reached in $s^\mathcal{F}$.

Theorem 7. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, s_\star \rangle$ be an FDR task and $\mathcal{F}$ an inverted-fork factoring where every leaf consists of a single variable, $\forall F^L \in \mathcal{F}^L : |F^L| = 1$. Let $s^\mathcal{F}$ be a solvable non-goal decoupled state, and let $\pi^\mathcal{F}$ be a decoupled plan for $s^\mathcal{F}$. Let $\mathcal{A}_s^\mathcal{F}$ be a SVIFDSSS for $s^\mathcal{F}$.*

Then $\mathcal{A}_s^\mathcal{F}$ contains a center action starting a permutation of $\pi^\mathcal{F}$.

Proof. Let π be a global plan for $\pi^\mathcal{F}$, and assume, without loss of generality, by Lemma 1, that π is past-maximal. Denote $\pi = \langle a_1, \ldots, a_n \rangle$. As above, let a_t be the starting action of $\pi^C[\pi^\mathcal{F}]$ in π, and denote $\pi^{past} := \langle a_1, \ldots, a_{t-1} \rangle$ and $\pi^{future} := \langle a_t, \ldots, a_n \rangle$. Then the following properties hold:

(a) There must be a center action shared between $\mathcal{A}_s^\mathcal{F}$ and π^{future}, i.e., $a \in \mathcal{A}_s^\mathcal{F} \cap \{a_t, \ldots, a_n\}$. Such an action must exist by the same argument as for (a) in the proof of Theorem 1. It must be a center action due to Proposition 1.

Given (a), let a_k be the first shared action, i.e., say that $a_k \in \mathcal{A}_s^\mathcal{F}$ and $\{a_t, \ldots, a_{k-1}\} \cap \mathcal{A}_s^\mathcal{F} = \emptyset$.

(b) a_k is reached in $s^\mathcal{F}$.

By the same argument as for (b) in the proof of Theorem 6 .

(c) a_k does not interfere with any of the center actions a_i in $\langle a_t, \dots, a_{k-1} \rangle$, where $pre(a_k) \parallel pre(a_i)$.

For the center actions in $\langle a_t, \dots, a_{k-1} \rangle$, this holds by the same argument as for (c) in the proof of Theorem 6.

(d) The leaf preconditions of a_k agree with those of all center actions a_i on $\langle a_t, \dots, a_{k-1} \rangle$; formally $pre(a_k)[\mathcal{V} \setminus F^C] \parallel pre(a_i)[\mathcal{V} \setminus F^C]$, $t \leq i < k$.

Assume for contradiction that there exists an action a_i where $pre(a_k)[\mathcal{V} \setminus F^C] \not\parallel pre(a_i)[\mathcal{V} \setminus F^C]$ preceding a_k on π^{future}. By (a) and (b) a_k is a reached center action. Thus, with Definition 15 (iv), $\mathcal{A}_s^{\mathcal{F}}$ contains all center actions a' where $pre(a_k)[\mathcal{V} \setminus F^C] \not\parallel pre(a')[\mathcal{V} \setminus F^C]$, invoking Proposition 1 to show that actions with unreached leaf preconditions can be discarded. But then, a_i is such an action and is therefore contained in $\mathcal{A}_s^{\mathcal{F}}$, in contradiction to a_k being the first shared action.

(e) No action on $\langle a_t, \dots, a_{k-1} \rangle$ affects a variable in $vars(pre(a_k)) \cap \bigcup_{F^L \in \mathcal{F}^L} F^L$.

Observe first that, due to the inverted-fork structure, every action that affects a variable in $vars(pre(a_k)) \cap \bigcup_{F^L \in \mathcal{F}^L} F^L$ must be a leaf action $a \in \mathcal{A}^L[F^L]$. Moreover, all $\mathcal{A}^L[F^L]$ are disjoint and $vars(pre(a)) \cup vars(\mathit{eff}(a)) = F^L$ for all $a \in \mathcal{A}^L[F^L]$ (because $|F^L| = 1$, for all $F^L \in \mathcal{F}^L$). Further, if $vars(pre(a_k)) \cap F^L \neq \emptyset$ then $F^L \subseteq vars(pre(a_k))$.

To prove the claim, we next show that there exists no action a_i on $\langle a_t, \dots, a_{k-1} \rangle$, with $t \leq i < k$, where $a_i \in \mathcal{A}^L[F^L]$ and $F^L \subseteq vars(pre(a_k))$. Assume for contradiction that such an a_i exists. From (b), we have that (1) a_k is reached in $s^{\mathcal{F}}$. In particular, its leaf precondition on $vars(pre(a_k)) \cap F^L$ is reached in $s^{\mathcal{F}}$. With (d), we have that (2) no center action on $\langle a_t, \dots, a_{k-1} \rangle$ has a competing leaf precondition, so in particular there exists no center action a_j, with $t \leq j < k$, s.t. $pre(a_k)[F^L] \not\parallel pre(a_j)[F^L]$.

Denote by $\pi^L_{\to k}$ the F^L-affecting leaf-action subsequence in π prior to a_k that includes a_i. Because π is a plan, $\pi^L_{\to k}$ is compliant with $\pi^C[s^{\mathcal{F}}] \circ \pi^C[\langle a_t, \dots, a_{k-1} \rangle]$. From (1) and (2), it follows that $\pi^L_{\to k}$ is also compliant with $\pi^C[s^{\mathcal{F}}]$. Given that by prerequisite a_i is in $\langle a_t, \dots, a_{k-1} \rangle$ this is in contradiction to the assumption that π is past-maximal.

(f) The leaf precondition of a_k is true at the end of π^{past}, i.e., in $s_0[\![\pi^{past}]\!]$.

This follows directly from (b) and (e), and the fact the π is a plan.

(g) a_k can be moved to the start of π^{future}. Precisely, $\pi' := \pi^{past} \circ \langle a_k, a_t, \dots, a_{k-1}, a_{k+1}, \dots, a_n \rangle$ is a plan for Π.

Given (b) and (f), a_k is applicable in $s_0[\![\pi^{past}]\!]$. With (c) a_k does not interfere with any center action a_i in $\langle a_t, \dots, a_{k-1} \rangle$ by the same argument as in the proof of Theorem 1. The only leaf actions it can interfere with are those affecting $vars(pre(a_k)) \cap \bigcup_{F^L \in \mathcal{F}^L} F^L$, but with (e) there are no such action on π^{future} prior to a_k, concluding the argument.

The claim now follows with the same concluding argument as in the proof of Theorem 1. □

A.2 Exponential Separations

Theorem 4. *Decoupled search is exponentially separated from SSS, and vice versa.*

Proof. Consider again our running example with a truck and N packages on a map with two locations A and B. There are only 3 reachable decoupled states: in $s_0^{\mathcal{F}}$, all packages can be at A or loaded (i.e., these are the reachable leaf states); driving to B, all packages can also be at B; driving back to A yields the same center state but a different pricing function. SSS, on the other hand, does not yield any pruning. In any state s, to make progress to the goal, $\mathcal{A}_s$ must include one applicable *load* or *unload* action; which interferes with the applicable *drive* action; which in turn interferes with all applicable *load/unload* actions.

Vice versa, consider the task family F from Gnad and Hoffmann (2019) with $\Pi^n = \langle \mathcal{V}^n, \mathcal{A}^n, s_0^n, s_\star^n \rangle$ as follows. $\mathcal{V}^n = \{v_1, \dots, v_n\}$, where $\mathcal{D}(v_i) = \{0, 1, 2\}$ for $1 \leq i \leq n$. The initial state is $s_0^n = \{v_1 = 0, \dots, v_n = 0\}$, the goal state is $s_\star^n = \{v_1 = 2, \dots, v_n = 2\}$. The actions are $\mathcal{A}^n = \{a_0, a_i^{12}, a_{ij}^{12} \mid 1 \leq i, j \leq n\}$ where $pre(a_0) = \{v_1 = 0, \dots, v_n = 0\}$ and $\mathit{eff}(a_0) = \{v_1 = 1, \dots, v_n = 1\}$; $pre(a_i^{12}) = \{v_i = 1\}$ and $\mathit{eff}(a_i^{12}) = \{v_i = 2\}$; $pre(a_{ij}^{12}) = \{v_i = 0, v_j = 1\}$ and $\mathit{eff}(a_{ij}^{12}) = \{v_i = 2, v_j = 2\}$.

First, a_0 is applied to s_0^n. For the successor state $s_1 := s_0^n[\![a_0]\!]$, SSS picks a variable v_i with unsatisfied goal, and adds the necessary enabling set actions a_i^{12}, a_{ij}^{12}, and a_{ji}^{12} to the stubborn set $\mathcal{A}_s$. Since no a_{kl}^{12} action is ever applicable, their only enabler a_0 is added to $\mathcal{A}_s$. Except a_i^{12}, no action in $\mathcal{A}_s$ is applicable in s_1, but all actions interfering with a_i^{12} are already in $\mathcal{A}_s$, so it is the only action applied to s_1. The same happens in all successors of s_1.

The decoupled state space is exponential in n as claimed. The a_{ij}^{12} actions have an unreachable precondition, yet their presence means that, in any star factoring, there can be at most one leaf: if there were two leaves F_i^L and F_j^L containing v_i and v_j respectively, then the action a_{ij}^{12} would incur a direct dependency across F_i^L and F_j^L. Thus, for any family $\mathcal{F}^n = \{F_n^C, F_n^L\}$ of star factorings (where F_n^L may not be present for some values of n), $\max(|F_n^C|, |F_n^L|) \in \Omega(n)$. So the number of reachable decoupled states is exponential in n since it has to enumerate all applications of a_i^{12} actions for a linear number of variables v_i. □

Theorem 5. *There exists a parameterized example family F such that, on F, DSSS yields an exponentially stronger reduction than both, decoupled search and SSS.*

Proof. Consider our example but with M trucks and $N * M$ packages, where each truck t_i is associated with a group of N packages that only t_i can transport (all trucks and packages start at A, all packages must be transported to B). The number of reachable decoupled states is exponential in M, because all trucks must be in the center factor, and their move combinations are enumerated. For SSS, as soon as $\mathcal{A}_s$ contains a *load/unload* action for one group of N packages, the *load/unload* actions for all other packages in that group are present as well, due to interference as before. So the SSS-pruned reachable state space has size exponential in N.

Consider now decoupled search with DSSS pruning. In $s_0^{\mathcal{F}}$, all packages can be at A or loaded into their respective truck. The necessary enabling set for $s_\star$ will select one package, associated with some truck t_i; hence $\mathcal{A}_s$ includes *drive*(t_i, A, B). This does not interfere with the *drive* actions for the other trucks, so it is the *only* applicable center action in $\mathcal{A}_s$, and we get a single successor state $s^{\mathcal{F}}$. In $s^{\mathcal{F}}$, the packages associated with t_i can all be at B. So the decoupled necessary enabling set for $s_\star$ for DSSS selects a package associated with another truck $t_j \neq t_i$. The only non-pruned

action is $\mathit{drive}(t_j, A, B)$; and so forth. Once all trucks are at B, we have a goal decoupled state $s^{\mathcal{F}}_\star$. The goal-price frontier sets for all $F^L \in \mathcal{F}^L$ in $s^{\mathcal{F}}_\star$ are empty, because the package prices are already the cheapest possible ones. So there are exactly $M + 1$ reachable decoupled states. □

References

Alkhazraji, Y., Wehrle, M., Mattmüller, R., & Helmert, M. (2012). A stubborn set algorithm for optimal planning. In Raedt, L. D. (Ed.), *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*, pp. 891–892, Montpellier, France. IOS Press.

Amir, E., & Engelhardt, B. (2003). Factored planning. In Gottlob, G. (Ed.), *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pp. 929–935, Acapulco, Mexico. Morgan Kaufmann.

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

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*(1–2), 5–33.

Brafman, R., & Domshlak, C. (2003). Structure and complexity in planning with unary operators. *Journal of Artificial Intelligence Research*, *18*, 315–349.

Brafman, R., & Domshlak, C. (2006). Factored planning: How, when, and when not. In Gil, Y., & Mooney, R. J. (Eds.), *Proceedings of the 21st National Conference of the American Association for Artificial Intelligence (AAAI'06)*, pp. 809–814, Boston, Massachusetts, USA. AAAI Press.

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

Brafman, R. I., & Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In Rintanen, J., Nebel, B., Beck, J. C., & Hansen, E. (Eds.), *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pp. 28–35. AAAI Press.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, *35*(8), 677–691.

Crosby, M., Rovatsos, M., & Petrick, R. P. A. (2013). Automated agent decomposition for classical planning. In Borrajo, D., Fratini, S., Kambhampati, S., & Oddi, A. (Eds.), *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy. AAAI Press.

Domshlak, C., Hoffmann, J., & Katz, M. (2015). Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, *221*, 73–114.

Domshlak, C., Katz, M., & Shleyfman, A. (2012). Enhanced symmetry breaking in cost-optimal planning as forward search. In Bonet, B., McCluskey, L., Silva, J. R., & Williams, B. (Eds.), *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press.

Edelkamp, S. (2001). Planning with pattern databases. In Cesta, A., & Borrajo, D. (Eds.), *Proceedings of the 6th European Conference on Planning (ECP'01)*, pp. 13–24. Springer-Verlag.

Edelkamp, S., & Helmert, M. (1999). Exhibiting knowledge in planning problems to minimize state encoding length. In Biundo, S., & Fox, M. (Eds.), *Proceedings of the 5th European Conference on Planning (ECP'99)*, pp. 135–147. Springer-Verlag.

Edelkamp, S., Leue, S., & Lluch-Lafuente, A. (2004). Partial-order reduction and trail improvement in directed model checking. *International Journal on Software Tools for Technology Transfer*, *6*(4), 277–301.

Fabre, E., Jezequel, L., Haslum, P., & Thiébaux, S. (2010). Cost-optimal factored planning: Promises and pitfalls. In Brafman, R. I., Geffner, H., Hoffmann, J., & Kautz, H. A. (Eds.), *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pp. 65–72. AAAI Press.

Fortune, S., Hopcroft, J., & Wyllie, J. (1980). The directed subgraph homeomorphism problem. *Theoretical Computer Science*, *10*(2), 111 – 121.

Franco, S., Lelis, L. H., & Barley, M. (2018). The complementary2 planner in the IPC 2018. In *IPC 2018 planner abstracts*.

Franco, S., Torralba, A., Lelis, L. H., & Barley, M. (2017). On creating complementary pattern databases. In Sierra, C. (Ed.), *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press/IJCAI.

Gaschnig, J. (1977). Exactly how good are heuristics?: Toward a realistic predictive theory of best-first search. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI'77)*, pp. 434–441, Cambridge, MA. William Kaufmann.

Gerevini, A., Saetti, A., & Serina, I. (2003). Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research*, *20*, 239–290.

Gnad, D., Dubbert, P., Lluch-Lafuente, A., & Hoffmann, J. (2018). Star-topology decoupling in SPIN. In del Mar Gallardo, M., & Merino, P. (Eds.), *Proceedings of the 25th International Symposium on Model Checking of Software (SPIN'18)*, Lecture Notes in Computer Science. Springer.

Gnad, D., & Hoffmann, J. (2015). Beating LM-cut with h^{max} (sometimes): Fork-decoupled state space search. In Brafman, R., Domshlak, C., Haslum, P., & Zilberstein, S. (Eds.), *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, pp. 88–96. AAAI Press.

Gnad, D., & Hoffmann, J. (2018). Star-topology decoupled state space search. *Artificial Intelligence*, *257*, 24 – 60.

Gnad, D., & Hoffmann, J. (2019). On the relation between star-topology decoupling and petri net unfolding. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*. AAAI Press.

Gnad, D., Hoffmann, J., & Domshlak, C. (2015). From fork decoupling to star-topology decoupling. In Lelis, L., & Stern, R. (Eds.), *Proceedings of the 8th Annual Symposium on Combinatorial Search (SOCS'15)*, pp. 53–61. AAAI Press.

Gnad, D., Poser, V., & Hoffmann, J. (2017a). Beyond forks: Finding and ranking star factorings for decoupled search. In Sierra, C. (Ed.), *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press/IJCAI.

Gnad, D., Torralba, Á., & Hoffmann, J. (2017b). Symbolic leaf representation in decoupled search. In Fukunaga, A., & Kishimoto, A. (Eds.), *Proceedings of the 10th Annual Symposium on Combinatorial Search (SOCS'17)*. AAAI Press.

Gnad, D., Torralba, Á., Shleyfman, A., & Hoffmann, J. (2017c). Symmetry breaking in star-topology decoupled search. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*. AAAI Press.

Gnad, D., Wehrle, M., & Hoffmann, J. (2016). Decoupled strong stubborn sets. In Kambhampati, S. (Ed.), *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, pp. 3110–3116. AAAI Press/IJCAI.

Godefroid, P., & Wolper, P. (1991). Using partial orders for the efficient verification of deadlock freedom and safety properties. In *Proceedings of the 3rd International Workshop on Computer Aided Verification (CAV'91)*, pp. 332–342.

Hall, D., Cohen, A., Burkett, D., & Klein, D. (2013). Faster optimal planning with partial-order pruning. In Borrajo, D., Fratini, S., Kambhampati, S., & Oddi, A. (Eds.), *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy. AAAI Press.

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

Helmert, M. (2009). Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, *173*, 503–535.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In Gerevini, A., Howe, A., Cesta, A., & Refanidis, I. (Eds.), *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pp. 162–169. AAAI Press.

Helmert, M., Haslum, P., Hoffmann, J., & 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., & Röger, G. (2008). How good is almost perfect?. In Fox, D., & Gomes, C. (Eds.), *Proceedings of the 23rd National Conference of the American Association for Artificial Intelligence (AAAI'08)*, pp. 944–949, Chicago, Illinois, USA. AAAI Press.

Hoffmann, J., Kissmann, P., & Torralba, Á. (2014). "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Schaub, T. (Ed.), *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, Prague, Czech Republic. IOS Press.

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

Jonsson, P., & Bäckström, C. (1995). Incremental planning. In *European Workshop on Planning*.

Katz, M., & Domshlak, C. (2008). Structural patterns heuristics via fork decomposition. In Rintanen, J., Nebel, B., Beck, J. C., & Hansen, E. (Eds.), *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pp. 182–189. AAAI Press.

Katz, M., & Domshlak, C. (2010). Implicit abstraction heuristics. *Journal of Artificial Intelligence Research*, *39*, 51–126.

Kelareva, E., Buffet, O., Huang, J., & Thiébaux, S. (2007). Factored planning using decomposition trees. In Veloso, M. (Ed.), *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pp. 1942–1947, Hyderabad, India. Morgan Kaufmann.

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

Lansky, A. L., & Getoor, L. (1995). Scope and abstraction: Two criteria for localized planning. In Mellish, S. (Ed.), *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pp. 1612–1619, Montreal, Canada. Morgan Kaufmann.

McDermott, D. V. (1999). Using regression-match graphs to control search in planning. *Artificial Intelligence*, *109*(1–2), 111–159.

Nissim, R., & Brafman, R. (2014). Distributed heuristic forward search for multi-agent planning. *Journal of Artificial Intelligence Research*, *51*, 293–332.

Nissim, R., Brafman, R. I., & Domshlak, C. (2010). A general, fully distributed multi-agent planning algorithm. In van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M., & Sen, S. (Eds.), *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pp. 1323–1330. IFAAMAS.

Pearl, J. (1984). *Heuristics*. Morgan Kaufmann.

Pochter, N., Zohar, A., & Rosenschein, J. S. (2011). Exploiting problem symmetries in state-based planners. In Burgard, W., & Roth, D. (Eds.), *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11)*, San Francisco, CA, USA. AAAI Press.

Richter, S., & Westphal, M. (2010). The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, *39*, 127–177.

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, *5*, 115–135.

Seipp, J., Pommerening, F., Sievers, S., & Helmert, M. (2017). Downward Lab. `https://doi.org/10.5281/zenodo.790461`.

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. (2016). Sympa: Symbolic perimeter abstractions for proving unsolvability. In *UIPC 2016 planner abstracts*, pp. 8–11.

Torralba, Á., Alcázar, V., Kissmann, P., & Edelkamp, S. (2017). Efficient symbolic search for cost-optimal planning. *Artificial Intelligence*, *242*, 52–79.

Torralba, Á., Gnad, D., Dubbert, P., & Hoffmann, J. (2016). On state-dominance criteria in fork-decoupled search. In Kambhampati, S. (Ed.), *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press/IJCAI.

Torralba, Á., & Hoffmann, J. (2015). Simulation-based admissible dominance pruning. In Yang, Q. (Ed.), *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pp. 1689–1695. AAAI Press/IJCAI.

Valmari, A. (1989). Stubborn sets for reduced state space generation. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, pp. 491–515.

Wang, D., & Williams, B. C. (2015). tburton: A divide and conquer temporal planner. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pp. 3409–3417. AAAI Press.

Wehrle, M., & Helmert, M. (2014). Efficient stubborn sets: Generalized algorithms and selection strategies. In Chien, S., Do, M., Fern, A., & Ruml, W. (Eds.), *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press.

Wehrle, M., Helmert, M., Alkhazraji, Y., & Mattmüller, R. (2013). The relative pruning power of strong stubborn sets and expansion core. In Borrajo, D., Fratini, S., Kambhampati, S., & Oddi, A. (Eds.), *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy. AAAI Press.