# Beyond Stars – Generalized Topologies for Decoupled Search Technical Report

**Daniel Gnad**[1, 2], **Álvaro Torralba**[3], **Daniel Fišer**[1, 4]
[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2]Linköping University, Sweden
[3]Aalborg University, Denmark
[4]Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic
daniel.gnad@liu.se; alto@cs.aau.dk; danfis@danfis.cz

## Abstract

Decoupled search decomposes a classical planning task by partitioning its variables such that the dependencies between the resulting factors form a star topology. In this topology, a single center factor can interact arbitrarily with a set of leaf factors. The leaves, however, can interact with each other only indirectly via the center. In this work, we generalize this structural requirement and allow arbitrary topologies. The components must not overlap, i. e., each state variable is assigned to exactly one factor, but the interaction between factors is not restricted. We show how this generalization is connected to star topologies, which implies the correctness of decoupled search with this novel type of decomposition. We introduce factoring methods that automatically identify these topologies on a given planning task. Empirically, the generalized factorings lead to increased applicability of decoupled search on standard IPC benchmarks, as well as to superior performance compared to known factoring methods.

## Introduction

Star-topology decoupled state-space search, decoupled search for short, tackles the state explosion problem by exploiting the dependency structure of the given model. In classical planning, decoupled search decomposes planning tasks by partitioning the state variables such that the dependencies between the resulting factors form a star topology (Gnad and Hoffmann 2018). Here, a single *center factor* $C$ can interact arbitrarily with the remaining set of *leaf factors* $\{L_1, \dots, L_n\}$. Thereby, decoupled search exploits a form of conditional independence of the leaves; given a sequence of *center actions*, i. e., actions that have an effect on the center, the leaves are independent. This allows for an enumeration of the *compliant leaf paths* for each leaf separately. The search is then performed over center actions only, where each sequence of center actions $\pi^C$ ends in a *decoupled state* consisting of a single *center state* (an assignment to $C$) and a non-empty set of *leaf states* (assignments to each $L_i$). The leaf states reached for an $L_i$ are exactly those $L_i$-assignments reachable by any sequence of $L_i$-actions that can be executed along $\pi^C$, i. e., that is *compliant* with $\pi^C$. Thus, a decoupled state compactly represents exponentially many explicit states, which share the same center state and result from all combinations of leaf states across leaf factors.

Application of decoupled search requires finding a suitable decomposition of the task into factors. Existing work has explored several methods to automatically find such a factoring. They partition the variables into factors by analyzing their causal dependencies (Gnad, Poser, and Hoffmann 2017; Gnad and Hoffmann 2018), or using integer linear programming (ILP) (Schmitt, Gnad, and Hoffmann 2019). All known approaches focus specifically on star-topology factorings with a designated center factor, where cross-leaf interactions are limited to actions affecting the center.

In this work, we generalize the concept of star factorings to *generalized factorings*, which allow arbitrary cross-factor interactions. In fact, we even allow factorings *without* center factor. Our main motivation for the generalization is that very few actions (even a single one) can render star factorings impossible if such actions lead to direct cross-leaf interactions. By relaxing the requirements of factorings, such actions can simply be made center actions, which are considered by the main search. We do so by extending the notion of center actions to those that share precondition or effect variables with more than one leaf factor. We prove the correctness of this novel form of decomposition by connecting it to the known star topologies, extending the possibilities of decoupled search while keeping all its desirable properties.

We illustrate the advantages of generalized factorings on a collaborative robotics task and devise a new ILP encoding with different optimization criteria to decompose planning tasks. Our evaluation in optimal and satisficing settings shows that generalized factorings open up a wide range of possibilities, enabling the use of decoupled search on planning tasks that could not be tackled before. While there is no single best strategy for every domain or setting, the factorings from our new strategies can significantly improve performance over prior methods and explicit-state search.

## Background

A *planning task* (Bäckström and Nebel 1995) is a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, where $\mathcal{V}$ is a finite set of *variables*, and each variable $v \in \mathcal{V}$ has a finite domain $\mathcal{D}(v)$. $\mathcal{A}$ is a finite set of *actions*. Each action $a \in \mathcal{A}$ is a triple $\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle$, where preconditions $\mathsf{pre}(a)$ and effects $\mathsf{eff}(a)$ are partial assignments to $\mathcal{V}$, and the cost

$\mathsf{cost}(a) \in \mathbb{R}^{0+}$ is a non-negative real number. A *state* is a complete assignment to $\mathcal{V}$, $I$ is the *initial state*, and the *goal* $G$ is a partial assignment to $\mathcal{V}$. For a partial assignment $p$, we denote by $vars(p) \subseteq \mathcal{V}$ the subset of variables on which $p$ is defined. For $V' \subseteq \mathcal{V}$, we denote the restriction of $p$ onto $V'$ by $p[V']$, i.e., the assignment to $V' \cap vars(p)$ by $p$. We identify (partial) states with sets of variable/value pairs. An action $a$ is *applicable* in a state $s$ if $\mathsf{pre}(a) \subseteq s$. Applying $a$ in a (partial) state $s$ changes the value of all $v \in vars(\mathsf{eff}(a)) \cap vars(s)$ to $\mathsf{eff}(a)[v]$, and leaves $s$ unchanged elsewhere. The outcome state is denoted $s[\![a]\!]$. A *plan* for $\Pi$ is an action sequence $\pi$ applicable in $I$ that ends in a state $s_G \supseteq G$; it is *optimal* if its summed-up action cost, denoted $\mathsf{cost}(\pi)$, is minimal among all plans for $\Pi$.

Decoupled search is a technique developed to avoid the combinatorial explosion of having to enumerate all possible variable assignments of causally independent parts of a planning task. It does so by partitioning the state variables $\mathcal{V}$ into a *star factoring*, whose elements are called *factors*.

**Definition 1 (Star Factoring)** *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G\rangle$ be a planning task. A* factoring *$\mathcal{F} \subset 2^{\mathcal{V}}$ for $\Pi$ is a partition of $\mathcal{V}$.*

*A pair $\mathcal{F}_s = \langle C, \mathcal{L}\rangle$ is a* star factoring *for $\Pi$, if $\{C\} \cup \mathcal{L}$ is a factoring and for all actions $a \in \mathcal{A}$ either there exists an $L \in \mathcal{L}$ such that $vars(\mathsf{pre}(a)) \subseteq C \cup L$ and $vars(\mathsf{eff}(a)) \subseteq L$, or $vars(\mathsf{eff}(a)) \cap C \neq \emptyset$. $C$ is called the* center factor *of $\mathcal{F}_s$, and $\mathcal{L}$ are its* leaf factors.

By imposing a structural requirement on the interaction between the factors, namely a *star topology*, decoupled search can efficiently handle cross-factor dependencies. Here, the center $C$ can interact arbitrarily with the leaves $\mathcal{L}$, but interaction between leaves is allowed only if the center is affected at the same time. Actions affecting $C$, i.e., with an effect on a variable in $C$, are called *center actions*, denoted $\mathcal{A}^C$, and those affecting a leaf are called *leaf actions*, denoted $\mathcal{A}^{\mathcal{L}}$. The actions that affect a particular leaf $L \in \mathcal{L}$ are denoted $\mathcal{A}^L$. We define the set of *leaf-only actions* of a leaf $L$ as $\mathcal{A}^L_{\not{C}} := \mathcal{A}^L \setminus \mathcal{A}^C$. A sequence of center actions applicable in $I$ in the projection onto $C$ is a *center path*, a sequence of $\mathcal{A}^L$-actions applicable in $I$ in the projection onto $L$, is a *leaf path*. A complete assignment to $C$, or to an $L \in \mathcal{L}$, is called a *center state*, or *leaf state*, respectively. $S^{\mathcal{L}}$ is the set of all leaf states and that of a particular leaf $L$ is denoted $S^L$.

A *decoupled state* $s^{\mathcal{F}}$ is a pair $\langle \mathsf{center}(s^{\mathcal{F}}), \mathsf{prices}(s^{\mathcal{F}})\rangle$ where $\mathsf{center}(s^{\mathcal{F}})$ is a center state, and $\mathsf{prices}(s^{\mathcal{F}}) : S^{\mathcal{L}} \mapsto \mathbb{R}^{0+} \cup \{\infty\}$ is a *pricing function*, mapping each leaf state to a non-negative *price*. By $\pi^C(s^{\mathcal{F}})$ we denote the center path that starts in the initial decoupled state $I^{\mathcal{F}}$ and ends in $s^{\mathcal{F}}$. The pricing function is maintained during decoupled search in a way so that the price of a leaf state $s^L$ is the cost of a cheapest leaf path that ends in $s^L$ and that is *compliant* with $\pi^C(s^{\mathcal{F}})$, i.e., that can be scheduled alongside the center path executed up to $s^{\mathcal{F}}$. A decoupled state $s^{\mathcal{F}}$ *satisfies* a condition $p$, a partial state, denoted $s^{\mathcal{F}} \models p$, iff (i) $p[C] \subseteq \mathsf{center}(s^{\mathcal{F}})$ and (ii) for every $L \in \mathcal{L}$ there exists an $s^L \in S^L$ s.t. $p[L] \subseteq s^L$ and $\mathsf{prices}(s^{\mathcal{F}})[s^L] < \infty$. We define the set of leaf actions *enabled* by a center state $s^C$ as $\mathcal{A}^{\mathcal{L}}|_{s^C} := \{a^L \mid a^L \in \mathcal{A}^{\mathcal{L}} \wedge \mathsf{pre}(a^L)[C] \subseteq s^C\}$. For a center state $s^C$ and a pair of leaf states $s^L_1, s^L_2 \in S^L$, by $c_{s^C}(s^L_1, s^L_2)$ we define the cost of a cheapest path of $\mathcal{A}^L_{\not{C}}|_{s^C}$ actions from $s^L_1$ to $s^L_2$. If no such path exists $c_{s^C}(s^L_1, s^L_2) = \infty$. A decoupled state $s^{\mathcal{F}}$ represents a set of explicit states, its *member states*, namely those states $s$ where $s^{\mathcal{F}} \models s$.

**Definition 2 (Decoupled State Space)** *Let $\Pi$ be a planning task, and $\mathcal{F} = \langle C, \mathcal{L}\rangle$ a star factoring for $\Pi$. The* decoupled state space *is a labeled transition system $\Theta^{\mathcal{F}}_{\Pi} = \langle \mathcal{S}^{\mathcal{F}}, \mathcal{A}^C, \mathsf{cost}|_{\mathcal{A}^C}, \mathcal{T}^{\mathcal{F}}, I^{\mathcal{F}}, \mathcal{S}^{\mathcal{F}}_G\rangle$ as follows:*

*(i) $\mathcal{S}^{\mathcal{F}}$ is the set of all decoupled states.*
*(ii) The transition labels are the center actions $\mathcal{A}^C$.*
*(iii) The cost function is that of $\Pi$, restricted to $\mathcal{A}^C$.*
*(iv) $\mathcal{T}^{\mathcal{F}}$ contains a transition $s^{\mathcal{F}} \xrightarrow{a^C} t^{\mathcal{F}} \in \mathcal{T}^{\mathcal{F}}$ whenever $a^C \in \mathcal{A}^C$ and $s^{\mathcal{F}}, t^{\mathcal{F}} \in \mathcal{S}^{\mathcal{F}}$ are such that:*
  1. *$\pi^C(s^{\mathcal{F}}) \circ \langle a^C\rangle = \pi^C(t^{\mathcal{F}})$,*
  2. *$s^{\mathcal{F}} \models \mathsf{pre}(a^C)$,*
  3. *$\mathsf{center}(s^{\mathcal{F}})[\![a^C]\!] = \mathsf{center}(t^{\mathcal{F}})$,*
  4. *for every leaf $L \in \mathcal{L}$ and leaf state $s^L \in S^L$, if $s^L \models \mathsf{pre}(a^C)[L]$, then $\mathsf{prices}(t^{\mathcal{F}})[s^L[\![a^C]\!]] = \mathsf{prices}(s^{\mathcal{F}})[s^L]$. Additionally, $\mathsf{prices}(t^{\mathcal{F}})[s^L] = \min_{t^L \in S^L} \mathsf{prices}(t^{\mathcal{F}})[t^L] + c_{\mathsf{center}(t^{\mathcal{F}})}(t^L, s^L)$.*

*(v) $I^{\mathcal{F}}$ is the* decoupled initial state, *where $\mathsf{center}(I^{\mathcal{F}}) := I[C]$, $\pi^C(I^{\mathcal{F}}) := \langle\rangle$, and, for every leaf $L \in \mathcal{L}$, $\mathsf{prices}(I^{\mathcal{F}})[I[L]] = 0$ and for all other leaf states $s^L \in S^L$, $\mathsf{prices}(I^{\mathcal{F}})[s^L] = c_{\mathsf{center}(I^{\mathcal{F}})}(I[L], s^L)$.*
*(vi) $\mathcal{S}^{\mathcal{F}}_G = \{s^{\mathcal{F}}_G \mid s^{\mathcal{F}}_G \models G\}$ are the* decoupled goal states.

Decoupled search runs a search over center actions only, enumerating, for each leaf separately, the set of leaf states that can be reached in the form of the pricing function. Every search algorithm and heuristic function can be employed on the decoupled state space (Gnad and Hoffmann 2018).

## Beyond Star Topologies

In this section, we generalize the concept of star factorings by introducing *generalized factorings*. The key novelty is that we no longer impose any structural restriction regarding the dependencies between factors, i.e., *any* partition of the state variables is a generalized factoring.

**Definition 3 (Generalized Factoring)** *Let $\Pi$ be a planning task. A pair $\mathcal{F}_g = \langle C, \mathcal{L}\rangle$ is a* generalized factoring *for $\Pi$, if either $\mathcal{L}$ or $\{C\} \cup \mathcal{L}$ is a factoring for $\Pi$. $C$ is the (possibly empty)* center factor *of $\mathcal{F}_g$, and $\mathcal{L}$ are its* leaf factors.

In addition to allowing arbitrary cross-factor interactions, we also consider empty center factors, i.e., factorings where all components are leaves. Note that every star factoring is also a generalized factoring, but not vice versa. Next, we resolve complications with respect to the interaction between factors by introducing the notion of *global actions*, which replace the center actions from star factorings.

**Definition 4 (Global Action)** *Let $\mathcal{F}_g = \langle C, \mathcal{L}\rangle$ be a generalized factoring for the planning task $\Pi$. An action $a \in \mathcal{A}$ is a* global action *iff there does not exist an $L \in \mathcal{L}$ such that $vars(\mathsf{pre}(a)) \subseteq C \cup L$ and $vars(\mathsf{eff}(a)) \subseteq L$. The set of all global actions is denoted $\mathcal{A}^G$.*

While leaf and leaf-only actions are defined as before, we need to adapt center actions to be able to handle more complex interactions. We call the new type of action *global* to make the distinction clear, since global actions not necessarily affect the center. Essentially, an action $a$ is global if it is not a leaf-only action of any leaf. In particular, actions with preconditions or effects on more than one leaf, but without center effect become global actions.

The basic concept of decoupled search remains intact. In Definition 2, we replace mentions to center actions $\mathcal{A}^C$ by global actions $\mathcal{A}^G$. The search then branches over global actions, and all leaf states reachable via leaf-only actions are enumerated for each leaf separately. The center state associated with a decoupled state can now be *empty* (if $C = \emptyset$). Hence, while we generalize the notion of factorings, the underlying idea of the decomposition is preserved. In particular, even though it might seem we adopt the flexibility of traditional factored planning approaches—and thus their weaknesses—our search still strictly distinguishes between center/global transitions where components interact, and leaf-only transitions that do not affect another leaf.

**Example 1** *We show the advantages of generalized factorings on a collaborative robotics task where $n$ battery-powered agents ($a_i$ variables) $move$ on a map. Moving consumes battery ($b_i$ variables), and an agent can $charge$ another one by sharing its battery charge. A single agent has a non-empty battery initially, all agents need to reach a goal location. Formally, the task is defined with variables $\mathcal{V} = \{a_1, b_1, \dots, a_n, b_n\}$, domains $\mathcal{D}(a_i) = \{l_1, \dots, l_m\}, \mathcal{D}(b_i) = \{0, \dots, B\}$, actions $\mathcal{A} = \{move(a_i, l_x, l_y, k), charge(a_i, a_j, l, x, y)\}$, where $i \neq j$, initial state $I = \{a_1 = l_1, b_1 = B\} \cup \{a_i = l_i, b_i = 0 \mid i > 1\}$, and goal $G = \{a_i = l_1\}$. The actions are defined as follows:* $\mathsf{pre}(move(a_i, l_x, l_y, k)) = \{a_i = l_x, b_i = k\}$, $\mathsf{eff}(move(a_i, l_x, l_y, k)) = \{a_i = l_y, b_i = k - 1\}$, *and* $\mathsf{pre}(charge(a_i, a_j, l, x, y)) = \{a_i = l, a_j = l, b_i = x, b_j = y\}$, $\mathsf{eff}(charge(a_i, a_j, l, x, y)) = \{b_i = x - 1, b_j = y + 1\}$.

*With generalized factorings, we can have every agent with its battery in a leaf $L_i = \{a_i, b_i\}$, the center is empty, resulting in a factoring where the number of leaves scales with the number of agents. The search then branches over the global $charge$ actions, the $move$ actions are leaf-only actions.*

*Due to the charge actions, however, by which any pair of agents can interact, in a star factoring we can only place a subset of the agents (and possibly their battery) into one leaf, and the remaining variables in the center. Thus, only a linear state-space reduction can be achieved by decoupled search. We remark that there actually exists a scaling star factoring (i. e., where $|\mathcal{L}| = n$), namely placing all battery variables in the center, and each agent in a separate leaf. As we will show in Proposition 1, though, this does not lead to* any *state-space reduction, since all actions are center actions.*

## Correctness

We prove the correctness of decoupled search with generalized factorings using a polynomial mapping from a task $\Pi$ and a generalized factoring $\mathcal{F}_g$ to a modified task $\Pi_s$ and a star factoring $\mathcal{F}_s$ such that the decoupled state spaces $\Theta_{\Pi}^{\mathcal{F}_g}$ for $\Pi$ and $\mathcal{F}_g$ and $\Theta_{\Pi_s}^{\mathcal{F}_s}$ for $\Pi_s$ and $\mathcal{F}_s$ are the same.

**Definition 5 (Star-Mapping)** *Let $\mathcal{F}_g = \langle C, \mathcal{L} \rangle$ be a generalized factoring for $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, and $\mathcal{A}^G$ the global actions for $\Pi$ and $\mathcal{F}_g$. $S(\Pi, \mathcal{F}_g) = \langle \Pi_s, \mathcal{F}_s \rangle$ is a* star mapping *constructed as follows: $\Pi_s := \langle \mathcal{V}_s, \mathcal{A}_s, I_s, G \rangle$ is the* star-mapped planning task *and $\mathcal{F}_s := \langle C_s, \mathcal{L} \rangle$ the* star-mapped factoring *for $\Pi_s$ given $\Pi$ and $\mathcal{F}_g$, where:*

- $\mathcal{V}_s := \mathcal{V} \cup \{x\}$*,* $x \notin \mathcal{V}$*, with* $\mathcal{D}(x) := \{0\}$*,*
- $\mathcal{A}_s := \{a_s \mid a \in \mathcal{A}^G, \mathsf{pre}(a_s) := \mathsf{pre}(a), \mathsf{cost}(a_s) := \mathsf{cost}(a), \mathsf{eff}(a_s) := \mathsf{eff}(a) \cup \{x = 0\}\} \cup (\mathcal{A} \setminus \mathcal{A}^G)$*,*
- $I_s := I \cup \{x = 0\}$*, and*
- $C_s := C \cup \{x\}$.

In words, we map a generalized factoring $\mathcal{F}_g$ for a task $\Pi$ to a star factoring $\mathcal{F}_s$ for a modified task $\Pi_s$ by introducing a new state variable $x$ with unary domain. Then $x$ is added to the center factor $C_s$ of $\mathcal{F}_s$ and we add an auxiliary effect $\{x = 0\}$ to all global actions. It is easy to see that $\mathcal{F}_s$ is indeed a valid star factoring for $\Pi_s$, where leaf-only actions $\mathcal{A}^L_{\not C}$ of a leaf $L \in \mathcal{L}$ affect only $L$ and are preconditioned by $C \cup L$, as for the generalized factoring. All other actions, namely the global actions of $\mathcal{F}_g$, are made center actions for $\mathcal{F}_s$ by adding the effect on $x$. Note that the set of plans for $\Pi$ is exactly the same as for $\Pi_s$, i. e., solutions are not affected.

For the proof of Theorem 1 we need to introduce some additional notation. We define a projection function $S^{\setminus x}$, which takes as input (partial) states and actions, and removes all occurrences of $x$ from these inputs. More precisely, for a (partial) state $s$ we have $S^{\setminus x}(s) = s[vars(s) \setminus \{x\}]$, and for an action $a$, we have $S^{\setminus x}(a) = S^{\setminus x}(\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle) = \langle \mathsf{pre}(a)[vars(\mathsf{pre}(a)) \setminus \{x\}], \mathsf{eff}(a)[vars(\mathsf{eff}(a)) \setminus \{x\}], \mathsf{cost}(a) \rangle$. We extend $S^{\setminus x}$ to sets of actions by applying it separately to every action contained in the set, and to decoupled states by applying it to all factor states.

**Theorem 1 (Correspondence)** *Let $\mathcal{F}_g$ be a generalized factoring for $\Pi$, let $S(\Pi, \mathcal{F}_g) = \langle \Pi_s, \mathcal{F}_s \rangle$, and let $\Theta_{\Pi}^{\mathcal{F}_g}$ and $\Theta_{\Pi_s}^{\mathcal{F}_s}$ be decoupled state spaces of $\mathcal{F}_g$ and $\mathcal{F}_s$, respectively. Then $\Theta_{\Pi}^{\mathcal{F}_g} = \Theta_{\Pi_s}^{\mathcal{F}_s} \setminus x$.*

**Proof:** Let $\Theta_{\Pi}^{\mathcal{F}_g} = \langle \mathcal{S}^{\mathcal{F}}_g, \mathcal{A}^G, \mathsf{cost}|_{\mathcal{A}^G}, \mathcal{T}^{\mathcal{F}}_g, I^{\mathcal{F}}_g, \mathcal{S}^{\mathcal{F}}_{G\,g} \rangle$ and $\Theta_{\Pi}^{\mathcal{F}_s} = \langle \mathcal{S}^{\mathcal{F}}_s, \mathcal{A}^C, \mathsf{cost}|_{\mathcal{A}^C}, \mathcal{T}^{\mathcal{F}}_s, I^{\mathcal{F}}_s, \mathcal{S}^{\mathcal{F}}_{G\,s} \rangle$ denote the respective decoupled state spaces.

By definition, we have $\mathcal{A}^G = S^{\setminus x}(\mathcal{A}^C)$, so $\mathsf{cost}|_{\mathcal{A}^G} = \mathsf{cost}|_{\mathcal{A}^C}$. For the initial states, let $I^{\mathcal{F}}_g = \langle \mathsf{center}(I^{\mathcal{F}}_g), \mathsf{prices}(I^{\mathcal{F}}_g) \rangle$, and $I^{\mathcal{F}}_s = \langle \mathsf{center}(I^{\mathcal{F}}_s), \mathsf{prices}(I^{\mathcal{F}}_s) \rangle$. Again, by definition, we have $\mathsf{center}(I^{\mathcal{F}}_g) = S^{\setminus x}(\mathsf{center}(I^{\mathcal{F}}_s)) = S^{\setminus x}(I[C] \cup \{x = 0\})$. For the pricing functions, observe that for every leaf $L \in \mathcal{L}$, the sets of leaf-only actions $\mathcal{A}^L_{\not C}$ for the two factorings coincide, so the same action sequences can be applied to the initial leaf states $I[L]$, which leads to the exact same pricing function $\mathsf{prices}(I^{\mathcal{F}}_g) = \mathsf{prices}(I^{\mathcal{F}}_s)$.

By a similar argument for arbitrary decoupled states $s^{\mathcal{F}}_g \in \mathcal{S}^{\mathcal{F}}_g$, we get that $s^{\mathcal{F}}_s \in \mathcal{S}^{\mathcal{F}}_s$, where $\mathsf{center}(s^{\mathcal{F}}_s) = \mathsf{center}(s^{\mathcal{F}}_g) \cup \{x = 0\}$ and $\mathsf{prices}(s^{\mathcal{F}}_s) = \mathsf{prices}(s^{\mathcal{F}}_g)$.

Let $s_g^{\mathcal{F}} \xrightarrow{a} t_g^{\mathcal{F}} \in \mathcal{T}_g^{\mathcal{F}}$ be a transition in $\Theta_\Pi^{\mathcal{F}_g}$. We next show that $s_s^{\mathcal{F}} \xrightarrow{a_s} t_s^{\mathcal{F}} \in \mathcal{T}_s^{\mathcal{F}}$ is a transition in $\Theta_\Pi^{\mathcal{F}_s}$, where $s_g^{\mathcal{F}} = S^{\setminus x}(s_s^{\mathcal{F}})$, $a = S^{\setminus x}(a_s)$, and $t_g^{\mathcal{F}} = S^{\setminus x}(t_s^{\mathcal{F}})$. Like just shown, the decoupled states $s_s^{\mathcal{F}}$ and $t_s^{\mathcal{F}}$ exist in $\mathcal{S}_s^{\mathcal{F}}$. As shown above, $a_s$ is a center action. It remains to show that (1) $a_s$ is applicable in $s_s^{\mathcal{F}}$ and that (2) applying it results in $t_s^{\mathcal{F}}$. Part (1) is easy to see, since $\mathsf{pre}(a_s) = \mathsf{pre}(a)$ by definition. For (2), observe that $\mathsf{eff}(a_s) = \mathsf{eff}(a) \cup \{x = 0\}$, so $\mathsf{center}(t_g^{\mathcal{F}}) \cup \{x = 0\} = \mathsf{center}(t_s^{\mathcal{F}})$. For the pricing function $\mathsf{prices}(t_s^{\mathcal{F}})$, we get, for all leaves $L \in \mathcal{L}$ and $s^L \in S^L$, whenever $\mathsf{prices}(t_g^{\mathcal{F}})[s^L] = \mathsf{prices}(s_g^{\mathcal{F}})[s^L]$ because $s^L \models \mathsf{pre}(a)[L]$, then, because the leaf preconditions and effects of $a_s$ equal those of $a$, it also holds that $\mathsf{prices}(t_s^{\mathcal{F}})[s^L] = \mathsf{prices}(s_s^{\mathcal{F}})[s^L]$. Finally, since the sets of leaf-only actions are the same for both factorings, for all leaf states $s^L \in S^L$, it holds that $\mathsf{prices}(t_s^{\mathcal{F}})[s^L] = \mathsf{prices}(t_g^{\mathcal{F}})[s^L]$.

Lastly, the set of decoupled goal states is not affected, i. e., $s_g^{\mathcal{F}} \in \mathcal{S}_{G\,g}^{\mathcal{F}}$ iff $s_s^{\mathcal{F}} \in \mathcal{S}_{G\,s}^{\mathcal{F}}$, where $\mathsf{center}(s_s^{\mathcal{F}}) = \mathsf{center}(s_g^{\mathcal{F}}) \cup \{x = 0\}$ and $\mathsf{prices}(s_s^{\mathcal{F}}) = \mathsf{prices}(s_g^{\mathcal{F}})$. □

With Theorem 1, we can perform decoupled search with generalized factorings like with star factorings, and all properties of decoupled search with star factorings are preserved.

**Corollary 1** *Decoupled search with generalized factorings is sound, complete, and preserves optimality.*

## Characteristics of Generalized Factorings

Gnad, Poser, and Hoffmann (2017) observed that the so called *mobility* is an important property of factorings, closely related to the state-space reduction of the decoupled search. For a generalized factoring $\mathcal{F}_g$, a leaf $L \in \mathcal{L}$ is *mobile* if there exists a leaf-only action for $L$, i. e., $|\mathcal{A}_{\not{C}}^L| > 0$. A factoring is mobile if all its leaves are. Next, we show that a state-space reduction is possible *only* with mobile leaves.

**Proposition 1 (Non-mobile Leaves)** *Let* $\Pi$ *be a planning task and* $\mathcal{F}_g = \langle C, \mathcal{L} \rangle$ *a generalized factoring for* $\Pi$. *If a leaf* $L \in \mathcal{L}$ *is not mobile, then in all decoupled states* $s^{\mathcal{F}}$ *reachable from the initial decoupled state* $I^{\mathcal{F}}$ *there exists exactly one* $s^L \in S^L$ *where* $\mathsf{prices}(s^{\mathcal{F}})[s^L] < \infty$.

**Proof:** The claim is true in the initial state, since there exists no leaf-only action for $L$, so $\mathsf{prices}(I^{\mathcal{F}})[I[L]] = 0$ and $\mathsf{prices}(I^{\mathcal{F}})[s^L] = \infty$ for all $s^L \neq I[L]$. Let $s^{\mathcal{F}}$ be a successor of $I^{\mathcal{F}}$ via global action $a^G$. Then the leaf state $s^L = I[L][\![a^C]\!]$ has a price of 0 in $s^{\mathcal{F}}$. Again, because there are no leaf-only actions of $L$, no other leaf state of $L$ is reached in $s^{\mathcal{F}}$. This argument applies inductively to all decoupled states reachable from $I^{\mathcal{F}}$. □

Consequently, a leaf $L \in \mathcal{L}$ for which there can only ever be a single reached leaf state can be merged into a new center factor $C \cup L$ without affecting the set of member states of any reachable decoupled state. In the extreme case where no leaf is mobile, every decoupled state has exactly one member state, so decoupled search degrades to explicit-state search.

While mobility can be seen as a qualitative property, prior work also tried to quantify the "amount of work a leaf can do on its own". We adopt the definition of mobility of a factoring as the sum of the number of leaf-only actions $|\mathcal{A}_{\not{C}}^L|$ of its leaves $L \in \mathcal{L}$. This definition allows us to reason about the reduction power of a factoring more accurately, since the number of leaf-only actions provides an estimate on how many leaf states can be reached in a single decoupled state.

Although the state-space reduction is exponential in the number of leafs, sometimes we can obtained stronger reduction by moving leaf variables into the center.

**Proposition 2** *Let* $\mathcal{F}_g = \langle C, \mathcal{L} \rangle$ *be a generalized factoring for* $\Pi$ *with mobility* $M$. *If there exist non-mobile leaves* $\{L_1, \dots L_n\} \subseteq \mathcal{L}$, *then there exists a mobile factoring with* $|\mathcal{F}_g| - n$ *leaves with mobility* $M' \geq M$.

**Proof:** The mobility of a leaf $L \in \mathcal{L}$ equals the number of leaf-only actions of $L$, namely $|\mathcal{A}_{\not{C}}^L|$. Thus, as $L_1, \dots, L_n$ are not mobile, the factoring $\mathcal{F}_g' = \langle C \cup L_1 \cup \dots \cup L_n, \mathcal{L} \setminus \{L_1, \dots, L_n\} \rangle$ has at least the mobility of $\mathcal{F}_g$.

We get a strictly higher mobility if there exists a leaf action $a \in \mathcal{A}^L$ of an $L \in \mathcal{L}$ where, w.l.o.g. for $L_1$, we have $vars(\mathsf{eff}(a)) \subseteq L$ and $vars(\mathsf{pre}(a)) \subseteq L \cup L_1$. Then $a$ was a global action for $\mathcal{F}_g$, but is a leaf-only action for $\mathcal{F}_g'$. □

Given the construction in the proof, it can even make sense to sacrifice a leaf with low mobility and merge it into the center, since this can increase the number of leaf-only actions for potentially many other leaves.

## Relation to Other Forms of Factored Planning

Decoupled search on star topologies differs conceptually from other factored planning approaches by requiring a specific structure to be present in a planning task. With generalized factorings, this is no longer the case. Like other factored planning methods, this allows arbitrary variable partitions.

In localized factored planning (e. g. Amir and Engelhardt 2003; Fabre et al. 2010; Brafman and Domshlak 2006, 2008, 2013), the planning process performs a local search for the individual factors. Solutions to these sub-problems are coordinated by resolving cross-factor interactions using a global constraint-satisfaction problem. Hierarchical factored planning (e. g. Knoblock 1994; Kelareva et al. 2007; Wang and Williams 2015) refines top-level solutions for an increasing number of factors while moving down a hierarchy of decreasing level of abstraction, resolving inconsistencies by backtracking to higher levels. The crucial difference to decoupled search is that keeping global (center) and leaf actions separate allows to perform a monolithic search, just over the more complex decoupled state space. Thus, there is never the need to coordinate sub-solutions, or backtrack from partial plans that cannot be refined. This also allows us to use any standard search algorithm, planning heuristic, or pruning method, which are orthogonal to the decomposition performed by decoupled search (Torralba et al. 2016; Gnad et al. 2017; Gnad, Hoffmann, and Wehrle 2019).

## Obtaining Generalized Factorings

Following prior work (Schmitt, Gnad, and Hoffmann 2019), we formulate the problem of finding a generalized factoring as an integer linear program (ILP) and describe several objective functions targeting different factoring properties.

First, we choose a set of candidates for leaves called *potential leaves*. Then we set constraints so that the resulting partition of variables is indeed a generalized factoring. Finally, we add more constraints ensuring that all leaves are mobile, as we have shown in Proposition 1 and 2 that non-mobile leaves do not contribute to the state-space reduction.

Before getting to the specifics of the ILP encoding, we define the notion of *action variable schemas* expressing causal dependencies between the center and leaves.

**Definition 6** *Given an action $a \in \mathcal{A}$, an* action variable schema *(or* a-schema *for short), denoted by $A_a$, is a tuple $A_a = \langle \mathsf{pre}(A_a), \mathsf{eff}(A_a) \rangle$, where $\mathsf{pre}(A_a) = vars(\mathsf{pre}(a))$, and $\mathsf{eff}(A_a) = vars(\mathsf{eff}(a))$. The set of all a-schemas is denoted by $\mathcal{K} = \{A_a \mid a \in \mathcal{A}\}$. Given a set of variables $L \subseteq \mathcal{V}$, $\mathcal{K}(L) = \{A \in \mathcal{K} \mid \mathsf{eff}(A) \subseteq L\}$ denotes a set of a-schemas affecting L. And given an a-schema $A \in \mathcal{K}$, $|A|$ denotes the number of actions with a-schema A.*

Besides the planning task $\Pi$, the ILP encoding depends on the choice of a set of potential leaves, $\Lambda \subseteq 2^{\mathcal{V}}$. We choose potential leaves as the sets of variables affected by at least one action, i. e., $\Lambda = \{\mathsf{eff}(A) \mid A \in \mathcal{K}\}$. Clearly, a set of variables $V$ such that $\mathsf{eff}(A) \not\subseteq V$ for every $A \in \mathcal{K}$ can never be a mobile leaf, which makes our selection of $\Lambda$ the set of minimal possible variable sets that can become mobile leaves. We experimented with supersets of these potential leaves, e. g., by combining pairs of intersecting or causally related leaves. However, this lead to a significant increase in the size of the encoding and never paid off in practice.

Our ILP encoding uses the following binary variables:

(i) $X_L$ for every potential leaf $L \in \Lambda$; $X_L$ is set to 1 when $L$ becomes a leaf, and it is set to 0 otherwise.

(ii) $Y_{L,A}$ for every potential leaf $L \in \Lambda$ and every a-schema $A \in \mathcal{K}(L)$. This variable is set to 1 when the actions $a \in \mathcal{A}$ with the a-schema $A$ are mobile for $L$.

(iii) $Z_v$ for every variable $v \in \mathcal{V}$. This variable is set to 1 if $v$ becomes a center variable in the resulting factoring.

The $X_L$ variables correspond to the actual choice of the variable partition that is made, $Y_{L,A}$ and $Z_v$ are used to ensure that we obtain a proper factoring.

The following constraints ensure that the solution to our ILP is indeed a mobile generalized factoring. There are constraints $X_L + X_{L'} \leq 1$ for every $L, L' \in \Lambda$ s.t. $L \neq L'$ and $L \cap L' \neq \emptyset$, enforcing a partition of the leaf variables. Constraints $Z_v \leq 1 - X_L$ for every $L \in \Lambda$ and every $v \in L$, ensure that the center is disjoint with all leaves.

A necessary condition for a leaf $L$ to be mobile is that there exists an action $a$ such that $vars(\mathsf{eff}(a)) \subseteq L$. So, it is enough to consider a-schemas $A$ with $\mathsf{eff}(A) \subseteq L$, i.e., the set $\mathcal{K}(L)$. For every potential leaf $L \in \Lambda$ we add the constraint:

$$X_L - \sum_{A \in \mathcal{K}(L)} Y_{L,A} \leq 0 \tag{1}$$

The next constraint serves two purposes. First, it ensures that any variable $Y_{L,A}$ is set to 1 only if the correponding variable $X_L$ is set to 1. Second, in combination with the previous constraint it enforces mobility of each leaf. The mobility of a leaf $L$ requires that there exists at least one action $a$ such that (a) $vars(\mathsf{eff}(a)) \in L$, and (b) every outside precondition of $a$ (i.e., $vars(\mathsf{pre}(a)) \setminus L$) is part of the center. As the constraint Eq. (1) takes care of (a), the following constraint ensures that (b) holds as well. For every potential leaf $L \in \Lambda$ and every action schema $A \in \mathcal{K}(L)$, we define the following constraint:

$$|\mathsf{pre}(A) \setminus L| \cdot Y_{L,A} - \sum_{v \in \mathsf{pre}(A) \setminus L} Z_v \leq X_L - Y_{L,A} \tag{2}$$

Finally, the following set of constraints makes sure that if a variable $v$ is not part of any leaf, then it is part of the center: $1 - \sum_{L \in \Lambda, v \in L} X_L \leq Z_v$ for every variable $v \in \mathcal{V}$.

Every solution to the ILP described above is a generalized factoring with mobile leaves. Next, we discuss different optimization criteria that can be applied.

## Towards Finding Good Factorings

Our main target is to obtain factorings that reduce the state-space the most, i. e., where each decoupled state has as many member states as possible. We consider several objective functions that correlate positively with this metric.

**Maximize number of mobile leaves** (L) This is desirable, as the number of member states is exponential in the number of leaves. In the ILP, the objective is to maximize $\sum X_L$.

**Maximize leaf mobility** (M) An increased leaf mobility contributes linearly to the state-space reduction. In the ILP, the objective is to maximize $\sum Y_{L,A}|A|$.

Similar to previous attempts to maximize mobility, though, this completely ignores how much each leaf contributes to the overall mobility. Ideally, we want a balanced contribution to maximize the search-space reduction. We capture this with the new notion of balanced leaf mobility.

**Maximize balanced leaf mobility** We define balanced leaf mobility as $\prod_{L \in \mathcal{L}} |\mathcal{A}^L_{\not\mathcal{C}}|$ or equivalently $\sum_{L \in \mathcal{L}} \log(|\mathcal{A}^L_{\not\mathcal{C}}|)$. That is, we multiply the number of mobile actions per leaf so that factorings with a balanced number of leaf-only actions per leaf are preferred (e.g., we prefer two leaves with 5 leaf-only actions each over one with 9 and another with only 1). In the ILP encoding, we can achieve this by introducing a binary variable $W_{L,C}$ for every combination $C \in 2^{\mathcal{K}(L)}$ of a-schemas from $\mathcal{K}(L)$ that can be mobile for a potential leaf $L$. We add constraints such that at most one $W_{L,C}$ can be set to 1 for every $L$. The objective is to maximize $\sum_{L \in \Lambda} \log(\sum_{A_i \in C} |A_i|) W_{L,C}$.

This encoding is not practical, though, as its size is exponential in the number of a-schemas of a potential leaf. Empirically, we observed an increase in the ILP size by one to two orders of magnitude on average, making it prohibitively large (statistics are available in the TR). Thus, we introduce an approximation that does not require additional variables.

**Maximize (approximate) balanced leaf mobility** (bM) For every potential leaf $L$, we divide the set of affecting actions $\mathcal{A}^L := \{a \in \mathcal{A} \mid vars(\mathsf{eff}(a)) \subseteq L\}$ into those whose precondition is contained in $L$, $\mathcal{A}^L_\top := \{a \in \mathcal{A}^L \mid vars(\mathsf{pre}(a)) \subseteq L\}$, and the rest, $\mathcal{A}^L_{pre} := \mathcal{A}^L \setminus \mathcal{A}^L_\top$. Note that, if $L$ is chosen as leaf factor, all actions in $\mathcal{A}^L_\top$ will be leaf-only actions, regardless of what variables are in the center. The remaining actions $\mathcal{A}^L_{pre}$ will be leaf-only actions iff $vars(\mathsf{pre}(a)) \subseteq L \cup C$. Hence, the minimum and maximum balanced mobility of $L$ are $M^L_{min} = \log(|\mathcal{A}^L_\top|)$ and $M^L_{max} = \log(|\mathcal{A}^L|)$, respectively. Moreover, let $\mathcal{A}^L_{pre}|_A$ denote the actions that have a-schema $A$ and are in $\mathcal{A}^L_{pre}$. Our objective function maximizes the following expression:

$$\sum_{L\in\Lambda} M^L_{min} X_L + \sum_{A\in\mathcal{K}(L)} Y_{L,A}(M^L_{max} - M^L_{min}) \frac{|\mathcal{A}^L_{pre}|_A|}{|\mathcal{A}_{pre}{}^L|}$$

The intuition is that the approximation is correct at the extremes, whenever none or all actions in $\mathcal{A}^L_{pre}$ are leaf-only actions. This is always the case if $|\mathcal{K}(L)| \leq 1$, which happens in many domains. In the middle, the function grows with the number of leaf-only actions, so that factorings with more leaf-only actions are always preferred.

**Maximize average leaf fact flexibility** (F) Prior definitions of mobility attempt to maximize the number of leaf-only actions, but ignore their effects. Yet, this is important because the number of member states of a decoupled state is exponential only in the number of leaves with more than one reached leaf state. If all leaf-only actions have exactly the same effect, then the number of leaf states with finite price is limited. Thus, we aim to maximize the percentage of actions affecting each leaf fact.

This can be encoded in the ILP by introducing a real variable $W_{v=d}$ with domain $[0,1]$ for every fact $v = d$ of $\Pi$, where $v \in \mathcal{V}$ and $d \in \mathcal{D}(v)$. The value of each $W_{v=d}$ corresponds to the percentage of actions with effect $v = d$ that are leaf-only actions. To set the value of $W_{v=d}$, we introduce a constraint $W_{v=d} = (\sum_{L\in\Lambda, A\in\mathcal{K}(L)} |A_{v=d}|Y_{L,A})/|\mathcal{A}_{v=d}|$, where $A_{v=d}$ is the subset of actions $a$ with a-schema $A$ where $\mathsf{eff}(a)[v] = d$, and $\mathcal{A}_{v=d}$ is the set of all actions with that effect. The objective is to maximize $\sum W_{v=d}$. We remark that this ignores the distribution of facts across leaves.

**Constraint on minimum leaf flexibility** While the previous objectives aim to maximize overall mobility across all leaves, here we target a minimum amount of mobility per leaf. Given a parameter $f_{min}$, we restrict the minimum flexibility of a potential leaf $L$ by introducing an additional constraint per candidate leaf $L \in \Lambda$:

$$f_{min} \cdot X_L \leq \sum_{A\in\mathcal{K}(L)} \frac{|A|Y_{L,A}}{|\{a \in \mathcal{A} \mid \mathsf{eff}(a) \cap L \neq \emptyset\}|} \quad (3)$$

Here, to select $L$ as a leaf, the ratio of leaf-only actions affecting $L$ must be at least $f_{min}$. As soon as $f_{min} > 0$, this is strictly more constrained than Equation 1, requiring not only that the leaf is mobile but that at least some minimum percentage of actions affecting it are leaf-only.

## Polynomial Test of Existence

The ILP can become very large for some tasks, so it is beneficial to check in advance if a non-trivial mobile factoring exists. There is an exact check that is quadratic in $|\mathcal{A}|$:

**Proposition 3 (Existence of Mobile 2-Leaf Factoring)** *There exists a mobile generalized factoring with at least two leaves iff there exist two distinct actions $a_1, a_2$ such that $vars(\mathsf{eff}(a_1)) \cap vars(\mathsf{eff}(a_2)) = \emptyset$, $vars(\mathsf{eff}(a_1)) \cap vars(\mathsf{pre}(a_2)) = \emptyset$, and $vars(\mathsf{pre}(a_1)) \cap vars(\mathsf{eff}(a_2)) = \emptyset$.*

**Proof:** Let $a_1, a_2$ be two actions such that $vars(\mathsf{eff}(a_1)) \cap vars(\mathsf{eff}(a_2)) = \emptyset$, $vars(\mathsf{eff}(a_1)) \cap vars(\mathsf{pre}(a_2)) = \emptyset$, and $vars(\mathsf{pre}(a_1)) \cap vars(\mathsf{eff}(a_2)) = \emptyset$. We construct a generalized factoring $\mathcal{F}_g$ with center $C$ and two leaves $L_1, L_2$ as follows: $L_1 = vars(\mathsf{eff}(a_1))$, $L_2 = vars(\mathsf{eff}(a_2))$, and $C = \mathcal{V} \setminus (L_1 \cup L_2)$. We next show that $\mathcal{F}_g$ is a proper generalized factoring with mobile leaves.

(i) By construction $\mathcal{F}_g$ is a partition of $\mathcal{V}$, so a generalized factoring. There are two non-empty leaf factors; the center $C$ is non-empty if $L_1 \cup L_2 \subset \mathcal{V}$.

(ii) Each leaf factor $L_i$ is mobile because (a) by construction there exists an action $a \in \mathcal{A}$ that affects only $L_i$, and because (b) $vars(\mathsf{pre}(a_1)) \cap vars(\mathsf{eff}(a_2)) = \emptyset$ there exists an action such that $vars(\mathsf{pre}(a)) \subseteq C \cup L_1$. The same argument also holds for $L_2$.

Let now $\mathcal{F}_g = \langle C, \mathcal{L}\rangle$ be a mobile factoring with center $C$ and at least two leaves $L_1 \neq L_2 \in \mathcal{L}$. Because $\mathcal{F}_g$ is mobile, there exists a leaf-only action $a_i$ for each of the leaves $L_i$ where $vars(\mathsf{pre}(a_i)) \subseteq C \cup L_i$ and $vars(\mathsf{eff}(a_i)) \subseteq L_i$. Thus, for $a_1$ and $a_2$ it holds that $vars(\mathsf{eff}(a_1)) \cap vars(\mathsf{eff}(a_2)) = \emptyset$, and $vars(\mathsf{eff}(a_i)) \cap vars(\mathsf{pre}(a_j)) = \emptyset$ for $\{i,j\} = \{1,2\}$. $\square$

With Proposition 3, we have an efficient way to check if a mobile 2-leaf factoring exists at all. We just need to iterate over all pairs of actions and check the stated conditions.

## Experimental Evaluation

We implemented our factoring strategies in the decoupled search planner of Gnad and Hoffmann (2018), which is based on Fast Downward (Helmert 2006). Our experiments were conducted using Lab (Seipp et al. 2017). We used all benchmarks of the International Planning Competitions (IPC) from 1998-2018 in the optimal as well as satisficing tracks, eliminating duplicate instances that appeared in several IPC iterations. For optimal planning, we report results for blind search and A$^*$ with $h^{\text{LM-cut}}$ (Helmert and Domshlak 2009); in satisficing planning, we use greedy best-first search (GBFS) with the $h^{\text{FF}}$ heuristic (Hoffmann and Nebel 2001), with and without preferred operator pruning (PO) using the common dual-queue approach (Richter and Helmert 2009). To compute these heuristics for decoupled states, prior work has introduced a task compilation that enables, in principle, the use of any heuristic. Via Theorem 1, this compilation is directly applicable to generalized factorings. The experiments were performed on a cluster of Intel E5-2660 machines running at 2.20 GHz with the runtime/memory limits of 30min/4GiB. Our code and data are publicly available (Gnad, Torralba, and Fišer 2022).

In our evaluation, we include the four ILP encodings that maximize the number of mobile leaves (L), the leaf mobility (M), the approximated balanced leaf mobility (bM), and

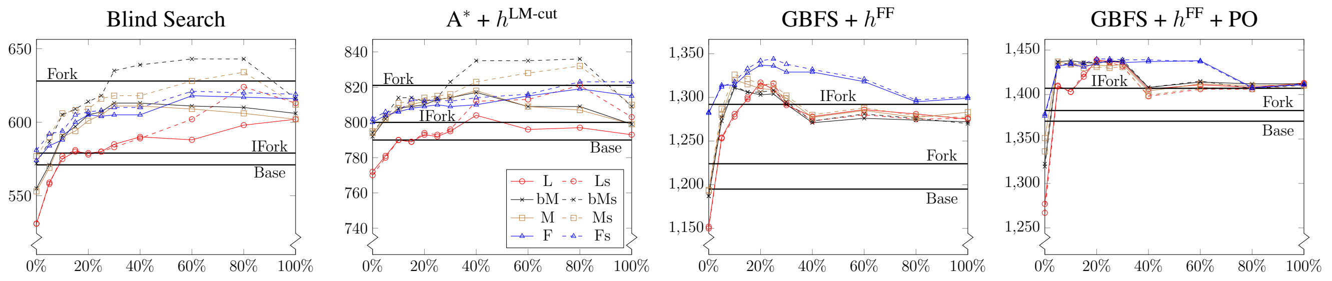

Figure 1: Coverage (on $y$-axis) of our new factoring strategies, with fallback to Base if a method abstains, as a function of the minimum leaf flexibility ($x$-axis). We also show the coverage of the baselines Base, Fork, and IFork.

the leaf-fact flexibility (F). For all these strategies, we run two variants. Our basic variant takes the set of all a-schema effects as potential leaves, the second one additionally considers the set of all strongly-connected components (SCC) in the causal graph (if there are at least two). We indicate the latter configurations with a postfix "s", e. g., Ls considers all a-schema effects and SCCs. We perform a systematic study of the influence of the minimum leaf flexibility on the performance of decoupled search. To do so, we run each of the aforementioned eight variants with increasing $f_{min}$: $0\% - 30\%$, in steps of 5, and $40\%, 60\%, 80\%, 100\%$. For non-zero flexibility, we indicate the used percentage in the configuration alias, e. g., bM60 maximizes balanced leaf mobility and enforces a minimum leaf flexibility of $60\%$.

We compare to explicit-state search (Base) and a set of existing factoring strategies for decoupled search as baselines. In particular, we run fork and inverted-fork factorings (Fork/IFork, in short Fo/IF), a strategy based on computing maximum independent sets of the causal graph (MIS), a greedy strategy (IA), and two ILP-based methods that produce *strict-star* (LPS), respectively general star factorings (LPG) (Gnad, Poser, and Hoffmann 2017; Schmitt, Gnad, and Hoffmann 2019). Strict-star factorings are a special case of star factorings where no action is allowed to touch, in precondition or effect, more than one leaf. For the ILP-based strategies, we include two variants in our evaluation, one that maximizes the number of mobile leaves, and another that maximizes leaf mobility. We only report data for the variants that maximize the mobility, since these give consistently better results. All baseline strategies (except the two that maximize mobility) have in common that they return the factoring with the maximum number of mobile leaves among the respective subset of considered factorings. For all factoring methods, we only consider candidate leaves with size (product of variable domain sizes) below $2^{32}$.

Like prior work on decoupled search, our methods *abstain* from solving a task if the generated factoring has less than two leaves. We say that a method *tackles* an instance if it does not abstain. We impose a runtime limit of 30s on the factoring process. On abstained instances, we sometimes (in particular in Figure 1) report the performance of the explicit-state baseline, because otherwise the underlying instance set would vary for each method. In this case, we consider an instance solved if the factoring method abstains or times out, and the baseline solves the instance in the remaining time.

In Figure 1, we show results of our minimum-leaf-flexibility investigation. For each of the eight base variants, we report the coverage (number of solved instances) as a function of the minimum flexibility for all four search settings. In addition to our new strategies, we also include the coverage of Base, Fork, and IFork for comparison. In general, the graphs confirm that the minimum flexibility has a significant impact on performance. For all methods, it is beneficial to impose at least a mild minimum of around $5\%$; and performance is worst without any restriction. Requiring a flexibility of $100\%$ almost never results in the highest coverage, either. There usually is a sweet-spot between the two extremes that yields the best performance. For satisficing planning, the data indicates a maximum coverage between $20\%$ and $40\%$, with a peak around $25\%$ for most methods. Adding the causal-graph SCCs does not lead to significant changes in satisficing planning. The SCCs do not seem to increase the number of alternative factorings that result in better performance in many cases. For optimal planning, the best coverage is achieved with higher minimum flexibility, between $40\%$ and $80\%$ for all but one configuration: blind search with L has its maximum at $100\%$. Adding SCCs is beneficial for all optimization criteria and results in significantly higher coverage for all but F.

Comparing the objective functions, there is a split into optimal and satisficing planning. For optimal planning bMs performs best, and Ms is good, too; for satisficing planning F is best. In all settings there are several configurations that outperform even the best previous method (which is either Fork or IFork), and all methods consistently beat Base with a minimum flexibility $> 5\%$.

In Table 1 we summarize the results for a selection of our new factoring strategies for $A^*$ search with $h^{\text{LM-cut}}$, and GBFS with $h^{\text{FF}}$ and PO. We show Base and all prior methods. From our strategies, we include the basic variant without enforced minimum flexibility and the best sub-configuration for all four objective functions. In addition, we show results for oracle configurations (Oracle(x), short O(x)), where $x \in \{old, new, dec, all\}$ denotes the set of configurations that are considered by the oracle: "old" are all previous strategies, "new" are the ones presented in this paper, "dec" considers both, and "all" includes Base as well. An oracle is a simulated best-case combination of all considered methods, which picks, per instance, the factoring that results in the lowest search time. For instances not solved

| | Optimal Planning: A$^*$ with LM-cut | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L80s | bM | bM80s | M | M80s | F | F80s | O(n) | O(d) | O(a) |
| Tackled | 1630 | 417 | 382 | 894 | 641 | 1124 | 1125 | 1243 | 1248 | 790 | 1232 | 789 | 1230 | 790 | 1236 | 864 | 1293 | 1321 | 1047 |
| Not t. before | - | - | - | - | - | - | - | - | 78 | 1 | 76 | 1 | 76 | 1 | 76 | 3 | 78 | 78 | 45 |
| # fork | - | 417 | 1 | 207 | 244 | 225 | 225 | 311 | 166 | 246 | 207 | 283 | 223 | 278 | 226 | 276 | 242 | 310 | 240 |
| # inv-fork | - | 0 | 381 | 136 | 46 | 137 | 57 | 193 | 30 | 231 | 29 | 233 | 34 | 240 | 85 | 166 | 107 | 179 | 173 |
| # strict-star | - | 0 | 0 | 551 | 351 | 762 | 274 | 625 | 40 | 199 | 35 | 195 | 20 | 202 | 26 | 155 | 91 | 544 | 431 |
| # star | - | 0 | 0 | 0 | 0 | 0 | 569 | 114 | 217 | 6 | 240 | 6 | 231 | 6 | 261 | 34 | 249 | 116 | 80 |
| # generalized | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 795 | 108 | 721 | 72 | 722 | 64 | 638 | 233 | 604 | 172 | 123 |
| $\bar{C} = \emptyset$ | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 3 | 1 | 1 | 3 | 2 | 2 | 0 | 0 |
| Solved | 790 | 266 | 132 | 475 | 397 | 549 | 567 | 638 | 585 | 423 | 597 | 438 | 599 | 434 | 606 | 480 | 672 | 681 | 850 |
| +Base | - | 821 | 800 | 803 | 802 | 813 | 810 | 837 | 772 | 821 | 792 | 836 | 794 | 832 | 800 | 823 | 837 | 840 | 850 |
| Solved\Base | - | 32 | 13 | 30 | 21 | 35 | 39 | 55 | 19 | 40 | 36 | 55 | 37 | 54 | 37 | 53 | 59 | 60 | 60 |
| Base\Solved | - | 1 | 3 | 17 | 9 | 12 | 19 | 8 | 37 | 9 | 34 | 9 | 33 | 12 | 27 | 20 | 12 | 10 | 0 |
| | Satisficing Planning: GBFS using FF and preferred-operator pruning | | | | | | | | | | | | | | | | | | |
| | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L20 | bM | bM30s | M | M5 | F | F20s | O(n) | O(d) | O(a) |
| Tackled | 1686 | 437 | 403 | 897 | 654 | 1173 | 1171 | 1309 | 1280 | 916 | 1270 | 896 | 1255 | 1093 | 1271 | 991 | 1335 | 1374 | 898 |
| Not t. before | - | - | - | - | - | - | - | - | 65 | 2 | 64 | 2 | 60 | 27 | 64 | 13 | 65 | 65 | 25 |
| # fork | - | 437 | 1 | 207 | 244 | 225 | 225 | 318 | 166 | 166 | 207 | 240 | 223 | 223 | 226 | 255 | 214 | 303 | 128 |
| # inv-fork | - | 0 | 402 | 96 | 46 | 140 | 41 | 219 | 9 | 143 | 19 | 191 | 26 | 121 | 97 | 156 | 166 | 183 | 169 |
| # strict-star | - | 0 | 0 | 594 | 364 | 808 | 277 | 587 | 32 | 222 | 51 | 176 | 23 | 120 | 31 | 137 | 156 | 435 | 251 |
| # star | - | 0 | 0 | 0 | 0 | 0 | 628 | 185 | 220 | 88 | 258 | 84 | 247 | 226 | 302 | 176 | 218 | 197 | 164 |
| # generalized | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 853 | 297 | 735 | 205 | 736 | 403 | 615 | 267 | 581 | 256 | 186 |
| $\bar{C} = \emptyset$ | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 1 | 2 | 0 | 1 | 0 |
| Solved | 1370 | 427 | 366 | 775 | 574 | 996 | 993 | 1167 | 1006 | 845 | 1046 | 832 | 1070 | 985 | 1102 | 899 | 1254 | 1279 | 1499 |
| +Base | - | 1382 | 1407 | 1383 | 1383 | 1361 | 1343 | 1440 | 1277 | 1439 | 1322 | 1439 | 1351 | 1437 | 1378 | 1440 | 1490 | 1493 | 1499 |
| Solved\Base | - | 12 | 37 | 35 | 19 | 43 | 79 | 104 | 35 | 82 | 66 | 81 | 71 | 98 | 90 | 91 | 128 | 129 | 129 |
| Base\Solved | - | 0 | 0 | 22 | 5 | 51 | 105 | 34 | 127 | 13 | 113 | 12 | 89 | 30 | 81 | 21 | 8 | 6 | 0 |

Table 1: Results summary on optimal and satisficing planning. See text for detailed explanations.

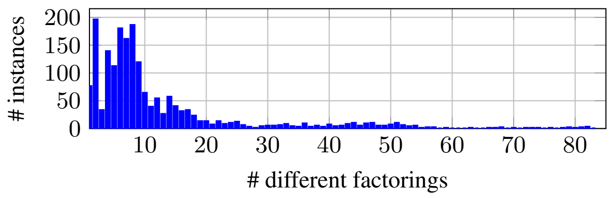

Figure 2: Distribution of different factorings per instance found by the 8 old and 88 new factoring methods.

by any method, it picks any of the methods that tackle the instance, and abstains only if all methods do.

The overall conclusions are similar for both settings. First, we observe that generalized factorings have opened up many new possibilities and they can tackle a lot more instances, up to 78 for $\mathrm{O(n)}$. This is largely due to generalized factorings, as confirmed by the second part of the table, which reports the kind of factorings found by each method. While, e. g., $\mathrm{Fork}$ and $\mathrm{IFork}$ obviously always return fork/inverted-fork factorings, $\mathrm{IA}$, $\mathrm{MIS}$, $\mathrm{LPS}$, and $\mathrm{LPG}$ produce a larger variety of star topologies. Our new strategies make great use of generalized no-star factorings, sometimes with empty center.

But are those new possibilities any good? Definitely, at least in some cases. Comparing the factorings preferred by the oracles, it turns out that strict-star factorings often yield the best performance. For both search settings these are by far preferred by $\mathrm{Oracle(all)}$, but generalized factorings are also performing better than all other factorings and the baseline in 123 instances for optimal and 186 for satisficing planning. The "+Base" row reveals that our best new methods outperform all prior ones, especially in satisficing planning. Some configurations, however, perform worse than $\mathrm{Base}$. Indeed, the best performing variants do not tackle that many new instances, showing that minimum leaf flexibility is a good criterion to decide when to use decoupled search. All strategies are capable to solve many instances not solved by the baseline (Solved\Base) and this is always larger when we enforce a minimum leaf flexibility. The other way around, there are often less instances that were tackled and not solved, but solved by $\mathrm{Base}$ (Base\Solved).

Figure 2 shows that there are many planning tasks for which our methods return a significant number of different factorings. The main peak is at 2 factorings per instance, but there exist tasks for which up to 82 different factorings were generated. Across both benchmark sets, with a total of 2330 instances, there are only 494 for which no factoring was computed, out of which in 340 no factoring with two mobile leaves is provably possible by Proposition 3.

Finally, in Figure 3 we report per-instance runtime results, comparing $\mathrm{Oracle(new)}$ to $\mathrm{Base}$ and $\mathrm{Oracle(old)}$. Every point below the diagonal indicates an instance where one of the new factoring strategies performs better than $\mathrm{Base}$ or than *every* of the existing methods. For blind search, the results look worst. Here, $\mathrm{Base}$ is often better than the new methods, though $\mathrm{Oracle(new)}$ actually solves 58 more instances. The old strategies perform close to the new ones, indicating that the oracles probably often chose the same

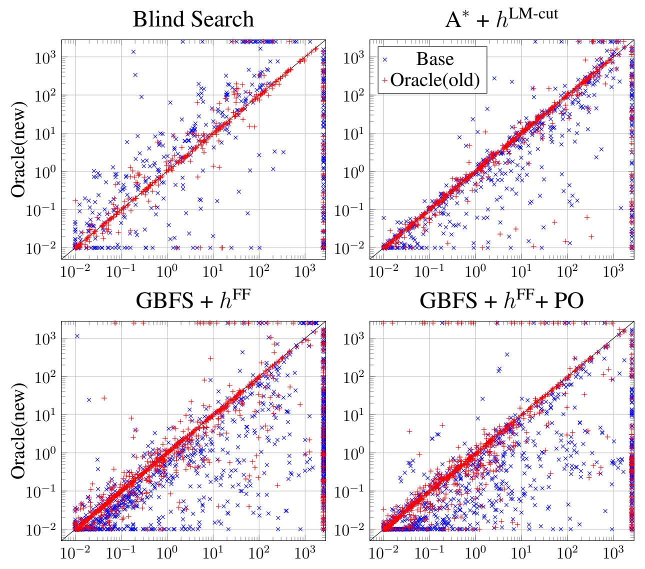

Figure 3: Per-instance runtime comparison for $\text{Oracle(new)}$ (on $y$-axis) vs. $\text{Base}$ and $\text{Oracle(old)}$ on the $x$-axis.

factorings. Still, there are instances where the new strategies result in a strong performance, sometimes several orders of magnitude faster than $\text{Base}$, and even than the best old method. For the remaining three search settings, the results look much more favorable for $\text{Oracle(new)}$. There are many cases where the performance is a lot better than $\text{Base}$ and $\text{Oracle(old)}$, and it happens only very rarely that the best of the new strategies performs a lot worse than both.

## Additional Results

In Tables 2–5 we summarize the results for a selection of our new factoring strategies for blind search, A$^*$ search with $h^{\text{LM-cut}}$, and GBFS using $h^{\text{FF}}$ with and without preferred operator pruning, respectively.

We will go through the tables from top to bottom, explaining the respective attributes along the discussion. For every configuration, we report the number of tackled instances; for the new strategies we also report the number of instances that were not tackled by any existing method. Here, we observe that some new strategies can tackle close to 80 additional instances (in optimal planning). The best performing variants, however, do not tackle many new instances. All methods return a very high number of unique factorings, factorings for an instance that were not also produced by another method. This number is, in comparison, fairly low for the new methods, as we consider all our different variants, which naturally produce similar factorings in many cases.

The "solved" rows reveal that decoupled search also solves many of the newly tackled instances, and, when falling back to $\text{Base}$ on abstained instances (row "+Base"), there are always variants that outperform all prior strategies. There are, however, even configurations that perform worse than $\text{Base}$ in this metric, which means that we would be better off not using decoupled search, at least in terms of total coverage. For all strategies, there is a fair number of instances solved that are not also solved by $\text{Base}$ ($\text{Solved}\setminus\text{Base}$). The other way around, there are usually significantly less instances that were tackled and not solved, but solved by $\text{Base}$ ($\text{Base}\setminus\text{Solved}$).

The subsequent rows show statistics on the number of special-case factorings obtained by each strategy. While, e. g., $\text{Fork}$ and $\text{IFork}$ obviously always return fork, respectively inverted-fork factorings, $\text{IA}$, $\text{MIS}$, $\text{LPS}$, and $\text{LPG}$ produce a large variety of topologies. The same is true for our new strategies, which often result in generalized factorings, sometimes with empty center factor. Comparing the oracle configurations, it turns out that strict-star factorings often yield the best performance. For both search settings these are by far preferred by $\text{Oracle(all)}$.

In the bottom part of the tables, we show the number of instances where a method performed best (with respect to FD's "search time") with an improvement factor of $X$ compared to all other methods. Here, a method is best for an instance if $X$ times its search time is smaller than the search time of all other methods that do not employ the same factoring. The main observation is that all methods solve instances at least 10 times faster than all others in a remarkable number of cases. On the other end, comparing the rows "Solved" and "Best 0.1" we can observe that most methods are at most a factor of 10 worse than any other method for most of the instances. The number of uniquely solved instances, i. e., only solved by that one method (for $\text{Base}$ we only consider instances tackled by at least one factoring strategy), indicates the complementarity of the strategies. For every single one of them, there are instances only solved by that method.

In Tables 6–9 we show per-domain results of the coverage (instances solved and falling back to $\text{Base}$ if the strategy abstains) for all four search settings. We include the same configurations as in Tables 2–5. Domains in which all configurations result in the same number of solved instances are summarized under "Others"; we highlight the highest coverage in **bold face**, and indicate domain/configuration pairs where decoupled search abstained on all instances by *italic font*.

In Table 10 we compare the average per-domain ILP encoding size (number of ILP variables and constraints) of the balanced leaf mobility ($\text{bMO}$) vs. the approximated balanced leaf mobility ($\text{bM}$). We show the size for our basic configurations that take the a-schema effects as potential leaves as well as the variant that includes the causal-graph SCCs. While the encoding size does not increase too much (or at all) in many domains when moving from the approximation to the exact encoding, there are also several exceptions where the size increases dramatically, up to the point where the ILPs cannot be solved within the given limits. Adding the causal-graph SCCs never leads to significant changes for the approximation. With the exact encoding, however, there are many domains in which the encoding size explodes.

## Conclusion

We have overcome the structural requirements of decoupled search and extended its applicability to planning tasks that could not be tackled before. The former restriction to star

| | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L80s | bM | bM60s | M | M80s | F | F60s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tackled | 1630 | 417 | 382 | 894 | 641 | 1125 | 1125 | 1242 | 1247 | 789 | 1232 | 815 | 1230 | 789 | 1235 | 875 | 1293 | 1320 | 994 |
| Not t. before | - | - | - | - | - | - | - | - | 77 | 1 | 76 | 1 | 76 | 1 | 76 | 3 | 78 | 78 | 48 |
| Unique $\mathcal{F}$ | - | 69 | 29 | 431 | 219 | 369 | 426 | - | 148 | 56 | 117 | 21 | 131 | 12 | 147 | 22 | - | - | - |
| Solved | 571 | 163 | 75 | 315 | 260 | 367 | 374 | 445 | 371 | 269 | 388 | 310 | 386 | 279 | 408 | 326 | 486 | 492 | 661 |
| +Base | - | 628 | 579 | 602 | 602 | 592 | 580 | 615 | 531 | 624 | 555 | 643 | 553 | 634 | 574 | 621 | 629 | 631 | 661 |
| Solved$\setminus$Base | - | 58 | 11 | 49 | 43 | 47 | 44 | 76 | 24 | 66 | 49 | 83 | 44 | 74 | 55 | 79 | 90 | 90 | 90 |
| Base$\setminus$Solved | - | 1 | 3 | 18 | 12 | 26 | 35 | 32 | 64 | 13 | 65 | 11 | 62 | 11 | 52 | 29 | 32 | 30 | 0 |
| # fork | - | 417 | 1 | 207 | 244 | 225 | 225 | 395 | 166 | 246 | 207 | 274 | 223 | 278 | 226 | 270 | 259 | 391 | 337 |
| # inv-fork | - | 0 | 381 | 136 | 46 | 137 | 57 | 189 | 30 | 231 | 29 | 216 | 35 | 240 | 85 | 117 | 97 | 188 | 163 |
| # strict-star | - | 0 | 0 | 551 | 351 | 763 | 274 | 574 | 40 | 198 | 35 | 170 | 20 | 201 | 26 | 154 | 89 | 527 | 370 |
| # star | - | 0 | 0 | 0 | 0 | 0 | 569 | 84 | 217 | 6 | 240 | 21 | 231 | 6 | 261 | 85 | 253 | 80 | 43 |
| # generalized | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 794 | 108 | 721 | 134 | 721 | 64 | 637 | 249 | 595 | 134 | 81 |
| $C = \emptyset$ | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 3 | 1 | 1 | 3 | 2 | 2 | 0 | 0 |
| Best 0.1 | 398 | 155 | 61 | 291 | 246 | 340 | 306 | 432 | 256 | 257 | 289 | 295 | 285 | 265 | 323 | 307 | 453 | 481 | 196 |
| Best 0.5 | 356 | 133 | 44 | 230 | 196 | 235 | 206 | 325 | 160 | 189 | 210 | 232 | 213 | 199 | 224 | 243 | 350 | 371 | 196 |
| Best 1.0 | 326 | 119 | 22 | 174 | 132 | 161 | 155 | 235 | 122 | 145 | 155 | 178 | 157 | 147 | 164 | 187 | 272 | 298 | 196 |
| Best 2 | 151 | 76 | 7 | 61 | 46 | 51 | 47 | 92 | 24 | 78 | 47 | 95 | 47 | 85 | 50 | 85 | 98 | 107 | 107 |
| Best 10 | 41 | 63 | 6 | 42 | 37 | 42 | 39 | 69 | 19 | 64 | 39 | 80 | 39 | 71 | 42 | 74 | 83 | 83 | 83 |
| Solved unique | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 1 | 11 | 1 | 11 | 3 | 13 | 14 | 14 | 14 |

Table 2: Results summary of optimal planning with blind search for explicit-state search (Base), existing factoring methods (left part), and our new strategies (right part). See text for detailed explanations.

topologies is obsolete, and we can now decompose planning tasks very flexibly by partitioning the state variables in an arbitrary way. We proved the correctness of the novel generalized factorings by making a connection to the existing star factorings. Thereby, all properties of decoupled search using star factorings are inherited, most importantly soundness, completeness, and optimality preservation.

We introduced a factoring strategy based on integer linear programming that is capable of producing generalized factorings, while being adaptable with respect to the properties of the factoring that should be optimized. Our experimental evaluation showed the effectiveness of our strategies and illustrated the benefit of the new freedom to decompose planning tasks for decoupled search. Given the vast space of possible factorings, however, the question of what the best decomposition for a given task is remains open. We believe this to be an important research topic for future work, that can further enhance the understanding, but also the performance of decoupled search.

## Acknowledgements

Daniel Gnad was supported by the German Research Foundation (DFG), under grant HO 2169/6-2, "Star-Topology Decoupled State Space Search". Jörg Hoffmann's research group has received support by DFG grant 389792660 as part of TRR 248 (see perspicuous-computing.science).

## References

Amir, E.; and Engelhardt, B. 2003. Factored Planning. In Gottlob, G., ed., *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, 929–935. Acapulco, Mexico: Morgan Kaufmann.

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

Brafman, R.; and Domshlak, C. 2006. Factored Planning: How, When, and When Not. In Gil, Y.; and Mooney, R. J., eds., *Proceedings of the 21st National Conference of the American Association for Artificial Intelligence (AAAI'06)*, 809–814. Boston, Massachusetts, USA: AAAI Press.

Brafman, R.; and Domshlak, C. 2013. On the Complexity of Planning for Agent Teams and Its Implications for Single Agent Planning. *Artificial Intelligence*, 198: 52–71.

Brafman, R. I.; and Domshlak, C. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, 28–35. AAAI Press.

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

Gnad, D.; and Hoffmann, J. 2018. Star-Topology Decoupled State Space Search. *Artificial Intelligence*, 257: 24 – 60.

Gnad, D.; Hoffmann, J.; and Wehrle, M. 2019. Strong Stubborn Set Pruning for Star-Topology Decoupled State Space Search. *Journal of Artificial Intelligence Research*, 65: 343–392.

Gnad, D.; Poser, V.; and Hoffmann, J. 2017. 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)*, 4310–4316. AAAI Press/IJCAI.

Gnad, D.; Torralba, Á.; and Fišer, D. 2022. Technical report and data of the ICAPS'22 paper "Beyond Stars - Generalized Topologies for Decoupled Search". https://doi.org/10.5281/zenodo.6384091.

| | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L80s | bM | bM80s | M | M80s | F | F80s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tackled | 1630 | 417 | 382 | 894 | 641 | 1124 | 1125 | 1243 | 1248 | 790 | 1232 | 789 | 1230 | 790 | 1236 | 864 | 1293 | 1321 | 1047 |
| Not t. before | - | - | - | - | - | - | - | - | 78 | 1 | 76 | 1 | 76 | 1 | 76 | 3 | 78 | 78 | 45 |
| Unique $\mathcal{F}$ | - | 69 | 29 | 429 | 219 | 369 | 429 | - | 150 | 56 | 120 | 12 | 136 | 13 | 150 | 14 | - | - | - |
| Solved | 790 | 266 | 132 | 475 | 397 | 549 | 567 | 638 | 585 | 423 | 597 | 438 | 599 | 434 | 606 | 480 | 672 | 681 | 850 |
| +Base | - | 821 | 800 | 803 | 802 | 813 | 810 | 837 | 772 | 821 | 792 | 836 | 794 | 832 | 800 | 823 | 837 | 840 | 850 |
| Solved$\setminus$Base | - | 32 | 13 | 30 | 21 | 35 | 39 | 55 | 19 | 40 | 36 | 55 | 37 | 54 | 37 | 53 | 59 | 60 | 60 |
| Base$\setminus$Solved | - | 1 | 3 | 17 | 9 | 12 | 19 | 8 | 37 | 9 | 34 | 9 | 33 | 12 | 27 | 20 | 12 | 10 | 0 |
| # fork | - | 417 | 1 | 207 | 244 | 225 | 225 | 311 | 166 | 246 | 207 | 283 | 223 | 278 | 226 | 276 | 242 | 310 | 240 |
| # inv-fork | - | 0 | 381 | 136 | 46 | 137 | 57 | 193 | 30 | 231 | 29 | 233 | 34 | 240 | 85 | 166 | 107 | 179 | 173 |
| # strict-star | - | 0 | 0 | 551 | 351 | 762 | 274 | 625 | 40 | 199 | 35 | 195 | 20 | 202 | 26 | 155 | 91 | 544 | 431 |
| # star | - | 0 | 0 | 0 | 0 | 0 | 569 | 114 | 217 | 6 | 240 | 6 | 231 | 6 | 261 | 34 | 249 | 116 | 80 |
| # generalized | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 795 | 108 | 721 | 72 | 722 | 64 | 638 | 233 | 604 | 172 | 123 |
| $C = \emptyset$ | - | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 3 | 1 | 1 | 3 | 2 | 2 | 0 | 0 |
| Best 0.1 | 581 | 261 | 115 | 454 | 390 | 525 | 535 | 629 | 533 | 410 | 547 | 432 | 551 | 426 | 571 | 461 | 667 | 676 | 417 |
| Best 0.5 | 474 | 238 | 92 | 411 | 354 | 482 | 465 | 603 | 425 | 372 | 457 | 397 | 459 | 388 | 485 | 422 | 651 | 672 | 417 |
| Best 1.0 | 274 | 150 | 71 | 230 | 189 | 263 | 234 | 381 | 190 | 219 | 212 | 243 | 214 | 234 | 230 | 252 | 464 | 540 | 417 |
| Best 2 | 19 | 50 | 11 | 42 | 37 | 50 | 49 | 67 | 18 | 56 | 41 | 73 | 43 | 71 | 41 | 67 | 78 | 81 | 81 |
| Best 10 | 15 | 37 | 1 | 23 | 23 | 25 | 25 | 40 | 14 | 37 | 23 | 49 | 24 | 47 | 24 | 46 | 50 | 51 | 51 |
| Solved unique | 10 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 3 | 1 | 4 | 1 | 4 | 1 | 3 | 5 | 5 | 5 |

Table 3: Results summary of optimal planning with A$^*$ using $h^{\text{LM-cut}}$ for explicit-state search (Base), existing factoring methods (left part), and our new strategies (right part). See text for detailed explanations.

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry Breaking in Star-Topology Decoupled Search. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, 125–134. AAAI Press.

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

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 162–169. AAAI Press.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Kelareva, E.; Buffet, O.; Huang, J.; and Thiébaux, S. 2007. Factored Planning Using Decomposition Trees. In Veloso, M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 1942–1947. Hyderabad, India: Morgan Kaufmann.

Knoblock, C. 1994. Automatically Generating Abstractions for Planning. *Artificial Intelligence*, 68(2): 243–302.

Richter, S.; and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 273–280. AAAI Press.

Schmitt, F.; Gnad, D.; and Hoffmann, J. 2019. Advanced Factoring Strategies for Decoupled Search using Linear Programming. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*. AAAI Press.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Torralba, Á.; Gnad, D.; Dubbert, P.; and 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)*, 3265–3271. AAAI Press/IJCAI.

Wang, D.; and Williams, B. C. 2015. tBurton: A Divide and Conquer Temporal Planner. In Bonet, B.; and Koenig, S., eds., *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 3409–3417. AAAI Press.

| | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L20 | bM | bM10s | M | M10s | F | F25s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tackled | 1686 | 437 | 403 | 897 | 656 | 1173 | 1171 | 1309 | 1280 | 916 | 1269 | 991 | 1257 | 992 | 1270 | 980 | 1335 | 1375 | 986 |
| Not t. before | - | - | - | - | - | - | - | - | 66 | 2 | 65 | 13 | 63 | 13 | 65 | 10 | 66 | 66 | 18 |
| Unique $\mathcal{F}$ | - | 85 | 27 | 474 | 233 | 405 | 494 | - | 153 | 91 | 148 | 79 | 168 | 71 | 176 | 44 | - | - | - |
| Solved | 1195 | 382 | 340 | 698 | 523 | 927 | 949 | 1088 | 924 | 787 | 955 | 831 | 959 | 845 | 1050 | 852 | 1182 | 1202 | 1404 |
| +Base | - | 1224 | 1292 | 1221 | 1217 | 1247 | 1250 | 1327 | 1152 | 1317 | 1187 | 1313 | 1193 | 1326 | 1282 | 1344 | 1383 | 1384 | 1404 |
| Solved$\setminus$Base | - | 29 | 99 | 42 | 28 | 103 | 136 | 169 | 79 | 136 | 95 | 135 | 106 | 143 | 161 | 159 | 206 | 209 | 209 |
| Base$\setminus$Solved | - | 0 | 2 | 16 | 6 | 50 | 80 | 37 | 121 | 14 | 102 | 17 | 107 | 12 | 73 | 10 | 18 | 20 | 0 |
| # fork | - | 437 | 1 | 207 | 244 | 225 | 225 | 243 | 166 | 166 | 207 | 216 | 223 | 248 | 226 | 257 | 229 | 264 | 141 |
| # inv-fork | - | 0 | 402 | 96 | 46 | 140 | 41 | 248 | 7 | 143 | 19 | 147 | 25 | 158 | 97 | 158 | 163 | 192 | 183 |
| # strict-star | - | 0 | 0 | 594 | 366 | 808 | 277 | 620 | 32 | 223 | 50 | 146 | 24 | 127 | 31 | 141 | 108 | 448 | 289 |
| # star | - | 0 | 0 | 0 | 0 | 0 | 628 | 198 | 220 | 87 | 258 | 202 | 247 | 184 | 302 | 167 | 266 | 214 | 177 |
| # generalized | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 855 | 297 | 735 | 280 | 738 | 275 | 614 | 257 | 569 | 257 | 196 |
| $C = \emptyset$ | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 1 |
| Best 0.1 | 848 | 345 | 317 | 631 | 482 | 846 | 871 | 1054 | 777 | 697 | 835 | 778 | 850 | 798 | 948 | 814 | 1172 | 1199 | 833 |
| Best 0.5 | 616 | 303 | 273 | 530 | 436 | 726 | 721 | 971 | 563 | 573 | 673 | 715 | 659 | 705 | 738 | 719 | 1122 | 1182 | 833 |
| Best 1.0 | 389 | 221 | 214 | 337 | 281 | 491 | 432 | 741 | 355 | 433 | 431 | 513 | 416 | 515 | 499 | 518 | 977 | 1116 | 833 |
| Best 2 | 40 | 39 | 22 | 39 | 36 | 52 | 70 | 115 | 27 | 60 | 62 | 72 | 56 | 79 | 61 | 81 | 166 | 187 | 187 |
| Best 10 | 23 | 27 | 2 | 18 | 17 | 20 | 39 | 52 | 16 | 38 | 36 | 37 | 36 | 48 | 36 | 48 | 67 | 72 | 72 |
| Solved unique | 20 | 0 | 0 | 0 | 0 | 2 | 2 | 3 | 9 | 4 | 13 | 10 | 9 | 10 | 10 | 9 | 40 | 40 | 40 |

Table 4: Results summary of satisficing planning with GBFS using $h^{\mathrm{FF}}$ without preferred-operator pruning for explicit-state search (Base), existing factoring methods (left part), and our new strategies (right part). See text for detailed explanations.

| | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L20 | bM | bM30s | M | M5 | F | F20s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tackled | 1686 | 437 | 403 | 897 | 654 | 1173 | 1171 | 1309 | 1280 | 916 | 1270 | 896 | 1255 | 1093 | 1271 | 991 | 1335 | 1374 | 898 |
| Not t. before | - | - | - | - | - | - | - | - | 65 | 2 | 64 | 2 | 60 | 27 | 64 | 13 | 65 | 65 | 25 |
| Unique $\mathcal{F}$ | - | 86 | 27 | 474 | 231 | 405 | 495 | - | 152 | 93 | 150 | 22 | 169 | 111 | 176 | 43 | - | - | - |
| Solved | 1370 | 427 | 366 | 775 | 574 | 996 | 993 | 1167 | 1006 | 845 | 1046 | 832 | 1070 | 985 | 1102 | 899 | 1254 | 1279 | 1499 |
| +Base | - | 1382 | 1407 | 1383 | 1383 | 1361 | 1343 | 1440 | 1277 | 1439 | 1322 | 1439 | 1351 | 1437 | 1378 | 1440 | 1490 | 1493 | 1499 |
| Solved$\setminus$Base | - | 12 | 37 | 35 | 19 | 43 | 79 | 104 | 35 | 82 | 66 | 81 | 71 | 98 | 90 | 91 | 128 | 129 | 129 |
| Base$\setminus$Solved | - | 0 | 0 | 22 | 5 | 51 | 105 | 34 | 127 | 13 | 113 | 12 | 89 | 30 | 81 | 21 | 8 | 6 | 0 |
| # fork | - | 437 | 1 | 207 | 244 | 225 | 225 | 318 | 166 | 166 | 207 | 240 | 223 | 223 | 226 | 255 | 214 | 303 | 128 |
| # inv-fork | - | 0 | 402 | 96 | 46 | 140 | 41 | 219 | 9 | 143 | 19 | 191 | 26 | 121 | 97 | 156 | 166 | 183 | 169 |
| # strict-star | - | 0 | 0 | 594 | 364 | 808 | 277 | 587 | 32 | 222 | 51 | 176 | 23 | 120 | 31 | 137 | 156 | 435 | 251 |
| # star | - | 0 | 0 | 0 | 0 | 0 | 628 | 185 | 220 | 88 | 258 | 84 | 247 | 226 | 302 | 176 | 218 | 197 | 164 |
| # generalized | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 853 | 297 | 735 | 205 | 736 | 403 | 615 | 267 | 581 | 256 | 186 |
| $C = \emptyset$ | - | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 1 | 2 | 0 | 1 | 0 |
| Best 0.1 | 936 | 381 | 341 | 689 | 534 | 923 | 908 | 1135 | 872 | 775 | 888 | 773 | 895 | 876 | 992 | 829 | 1238 | 1265 | 809 |
| Best 0.5 | 706 | 330 | 320 | 616 | 491 | 805 | 767 | 1029 | 690 | 689 | 723 | 706 | 744 | 769 | 823 | 741 | 1218 | 1252 | 809 |
| Best 1.0 | 476 | 300 | 257 | 466 | 384 | 615 | 535 | 820 | 514 | 552 | 534 | 576 | 543 | 592 | 607 | 574 | 1075 | 1185 | 809 |
| Best 2 | 33 | 16 | 32 | 49 | 31 | 22 | 29 | 69 | 24 | 47 | 35 | 58 | 29 | 46 | 30 | 45 | 118 | 127 | 127 |
| Best 10 | 20 | 14 | 18 | 33 | 16 | 15 | 15 | 39 | 14 | 34 | 17 | 32 | 17 | 33 | 17 | 33 | 42 | 45 | 45 |
| Solved unique | 6 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 6 | 2 | 7 | 2 | 7 | 4 | 10 | 8 | 25 | 25 | 25 |

Table 5: Results summary of satisficing planning with GBFS using $h^{\mathrm{FF}}$ and preferred-operator pruning for explicit-state search (Base), existing factoring methods (left part), and our new strategies (right part). See text for detailed explanations.

| Domain | # | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L80s | bM | bM60s | M | M80s | F | F60s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airport | 50 | **22** | ***22*** | ***22*** | ***22*** | 20 | 19 | 19 | 19 | 18 | ***22*** | 19 | ***22*** | 19 | ***22*** | 19 | 20 | 20 | 20 | **22** |
| DataNetwork | 20 | 6 | *6* | *6* | **9** | **9** | 7 | 6 | **9** | **9** | **9** | 7 | **9** | 6 | **9** | 8 | **9** | **9** | **9** | **9** |
| Depots | 22 | **4** | ***4*** | **4** | **4** | **4** | **4** | **4** | **4** | **4** | **4** | 3 | **4** | **4** | **4** | 3 | **4** | **4** | **4** | **4** |
| Driverlog | 20 | 7 | **11** | *7* | **11** | **11** | **11** | **11** | **11** | 6 | **11** | 10 | **11** | 9 | **11** | 10 | 10 | **11** | **11** | **11** |
| Elevators | 30 | 11 | *11* | **16** | **16** | 12 | **16** | 14 | **16** | 6 | **16** | 12 | **16** | 12 | 10 | **16** | 3 | **16** | **16** | **16** |
| Floortile | 40 | **2** | ***2*** | 0 | **2** | **2** | **2** | **2** | **2** | 0 | 0 | **2** | 0 | **2** | 0 | **2** | **2** | **2** | **2** | **2** |
| Freecell | 80 | **17** | ***17*** | ***17*** | 16 | ***17*** | 14 | 14 | 14 | 10 | ***17*** | 14 | ***17*** | 14 | ***17*** | 14 | ***17*** | 14 | 14 | **17** |
| GED | 20 | **15** | ***15*** | ***15*** | **15** | **15** | **15** | 13 | **15** | 13 | ***15*** | 13 | ***15*** | 13 | ***15*** | 13 | ***15*** | **15** | **15** | **15** |
| Hiking | 20 | **11** | ***11*** | ***11*** | ***11*** | ***11*** | **11** | **11** | **11** | 8 | **11** | 4 | **11** | 4 | **11** | 6 | **11** | **11** | **11** | **11** |
| Logistics | 63 | 12 | 27 | 13 | 26 | 26 | 27 | 27 | **28** | 12 | 19 | 27 | 27 | 27 | 27 | 27 | 27 | **28** | **28** | **28** |
| Maintenance | 5 | **5** | **5** | **5** | **5** | **5** | **5** | **5** | **5** | **5** | **5** | **5** | 4 | **5** | **5** | 4 | **5** | **5** | **5** | **5** |
| Miconic | 150 | 50 | **52** | *50* | **52** | 51 | 51 | 51 | **52** | **52** | 51 | 51 | **52** | 51 | **52** | 51 | 51 | **52** | **52** | **52** |
| Mprime | 35 | **19** | ***19*** | ***19*** | 17 | ***19*** | 14 | 14 | 14 | 8 | ***19*** | 9 | ***19*** | 8 | ***19*** | 10 | **19** | 15 | 17 | **19** |
| Mystery | 30 | **15** | ***15*** | **15** | **15** | ***15*** | **15** | 13 | 13 | 13 | **15** | 13 | **15** | 13 | **15** | 13 | **15** | 14 | 14 | **15** |
| NoMystery | 20 | 8 | **20** | *8* | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** |
| Openstacks | 80 | **31** | ***31*** | ***31*** | 26 | 28 | 26 | 24 | 28 | 24 | 26 | 24 | 26 | 24 | 26 | 24 | 26 | 26 | 28 | **31** |
| Organic-split | 20 | **10** | ***10*** | 9 | 7 | 9 | **10** | **10** | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 8 | 9 | 9 | **10** |
| ParcPrinter | 20 | 7 | *7* | *7* | 9 | *7* | 7 | 8 | 9 | 8 | **20** | 9 | **20** | 9 | **20** | 11 | **20** | **20** | **20** | **20** |
| Pathways | 30 | 4 | 4 | *4* | 4 | 4 | 4 | 4 | 4 | 4 | 4 | **5** | 4 | **5** | 4 | **5** | 4 | **5** | **5** | **5** |
| Pipesworld | 100 | **27** | ***27*** | ***27*** | ***27*** | ***27*** | 23 | 23 | 21 | 22 | ***27*** | 22 | ***27*** | 22 | ***27*** | 22 | ***27*** | 22 | 21 | **27** |
| PSR | 50 | 49 | 49 | *49* | **50** | **50** | 49 | 49 | **50** | 49 | 49 | 48 | 48 | 49 | 49 | **50** | **50** | **50** | **50** | **50** |
| Rovers | 40 | 5 | 6 | 6 | **7** | **7** | **7** | **7** | **7** | **7** | **7** | **7** | **7** | **7** | **7** | **7** | 6 | **7** | **7** | **7** |
| Satellite | 36 | 5 | 5 | 6 | 5 | 6 | **7** | **7** | **7** | 4 | 5 | **7** | 6 | **7** | 6 | **7** | 6 | **7** | **7** | **7** |
| Scanalyzer | 30 | **12** | ***12*** | ***12*** | 9 | 9 | 9 | 9 | 9 | 5 | ***12*** | 5 | ***12*** | 6 | ***12*** | 6 | ***12*** | 6 | 6 | **12** |
| Spider | 20 | **11** | ***11*** | ***11*** | ***11*** | 10 | **11** | **11** | 10 | **11** | ***11*** | **11** | ***11*** | **11** | ***11*** | **11** | ***11*** | **11** | 10 | **11** |
| Storage | 30 | **14** | ***14*** | ***14*** | ***14*** | ***14*** | ***14*** | 13 | 13 | 13 | ***14*** | 13 | ***14*** | 13 | ***14*** | 13 | ***14*** | 13 | 13 | **14** |
| Tetris | 17 | **9** | ***9*** | ***9*** | 6 | ***9*** | 8 | ***9*** | 6 | 6 | ***9*** | 6 | ***9*** | 6 | ***9*** | 6 | ***9*** | 6 | 6 | **9** |
| Tidybot | 30 | **16** | ***16*** | ***16*** | **16** | ***16*** | **16** | ***16*** | **16** | **16** | 14 | **16** | **16** | **16** | **16** | **16** | **16** | **16** | **16** | **16** |
| TPP | 30 | 6 | **24** | 6 | 6 | 6 | 6 | 6 | **24** | 5 | 16 | 5 | **24** | 5 | **24** | 5 | **24** | **24** | **24** | **24** |
| Transport | 59 | 17 | *17* | **18** | **18** | 17 | **18** | 14 | **18** | **18** | **18** | 11 | **18** | 11 | **18** | **18** | 7 | **18** | **18** | **18** |
| Trucks | 30 | **6** | ***6*** | ***6*** | 5 | **6** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | **6** |
| Woodworking | 30 | 7 | 8 | 9 | 8 | *7* | 10 | 10 | 11 | 9 | 9 | 10 | 10 | 10 | 9 | 11 | **13** | **13** | **13** | **13** |
| Zenotravel | 20 | 8 | **12** | 8 | 10 | 10 | 8 | 8 | **12** | 9 | **12** | 10 | **12** | 8 | 8 | 9 | **12** | **12** | **12** | **12** |
| Others | 353 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 | 123 |
| $\sum$ | 1630 | 571 | 628 | 579 | 602 | 602 | 592 | 580 | 615 | 531 | 624 | 555 | 643 | 553 | 634 | 574 | 621 | 629 | 631 | **661** |

Table 6: Per-domain coverage results (+Base) of the same configurations as shown in Table 2 with blind search. We use bold face to highlight highest coverage and italic font to indicate domains in which a decoupled search configuration abstained in all instances, so the number shown corresponds fully to Base.

| Domain | # | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L80s | bM | bM80s | M | M80s | F | F80s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airport | 50 | **28** | ***28*** | ***28*** | ***28*** | **28** | 27 | **28** | 27 | 27 | ***28*** | 27 | ***28*** | 27 | ***28*** | 27 | **28** | 27 | 27 | **28** |
| DataNetwork | 20 | 12 | *12* | *12* | **14** | **14** | 13 | 13 | **14** | **14** | **14** | 13 | **14** | 13 | **14** | **14** | **14** | **14** | **14** | **14** |
| Depots | 22 | **7** | ***7*** | **7** | **7** | **7** | **7** | **7** | **7** | 6 | **7** | 6 | **7** | 6 | **7** | 6 | **7** | **7** | **7** | **7** |
| Driverlog | 20 | 13 | 13 | *13* | 13 | 13 | **14** | **14** | **14** | 12 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | **14** | **14** |
| Elevators | 30 | 22 | *22* | **23** | **23** | 22 | **23** | **23** | **23** | 21 | **23** | 22 | **23** | 22 | 20 | **23** | 16 | **23** | **23** | **23** |
| Floortile | 40 | **13** | ***13*** | 11 | 8 | 8 | 8 | 2 | 11 | 3 | 11 | 2 | 11 | 2 | 11 | 2 | 11 | 11 | 11 | **13** |
| GED | 20 | **15** | ***15*** | ***15*** | **15** | **15** | **15** | 13 | **15** | 13 | ***15*** | 13 | ***15*** | 14 | ***15*** | **15** | ***15*** | **15** | **15** | **15** |
| Gripper | 20 | **7** | ***7*** | ***7*** | ***7*** | ***7*** | ***7*** | ***7*** | ***7*** | 6 | ***7*** | 6 | ***7*** | 6 | ***7*** | 6 | ***7*** | 6 | 6 | **7** |
| Hiking | 20 | 9 | *9* | *9* | *9* | *9* | **10** | **10** | **10** | 8 | **10** | 4 | **10** | 4 | **10** | 7 | **10** | **10** | **10** | **10** |
| Logistics | 63 | 26 | 34 | 27 | **35** | **35** | 34 | 34 | **35** | 22 | 29 | 34 | **35** | 34 | 34 | 34 | 34 | **35** | **35** | **35** |
| Miconic | 150 | **141** | 140 | ***141*** | 140 | 140 | 140 | 140 | **141** | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | **141** | **141** |
| Mprime | 35 | **22** | ***22*** | ***22*** | 20 | ***22*** | **22** | **22** | **22** | 21 | ***22*** | **22** | ***22*** | **22** | ***22*** | **22** | ***22*** | **22** | **22** | **22** |
| NoMystery | 20 | 14 | **20** | *14* | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** | **20** |
| Openstacks | 80 | **31** | ***31*** | ***31*** | 26 | 28 | 26 | 27 | 28 | 25 | 26 | 26 | 26 | 26 | 26 | 27 | 26 | 27 | 28 | **31** |
| Organic-split | 20 | **15** | ***15*** | 14 | 13 | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** | **15** |
| ParcPrinter | 20 | 10 | *10* | *10* | 13 | *10* | 14 | 17 | 17 | 16 | **20** | 18 | **20** | 18 | **20** | 18 | **20** | **20** | **20** | **20** |
| Parking | 40 | **5** | ***5*** | ***5*** | ***5*** | ***5*** | ***5*** | ***5*** | ***5*** | 3 | ***5*** | **5** | ***5*** | **5** | ***5*** | **5** | ***5*** | **5** | **5** | **5** |
| Pathways | 30 | **5** | **5** | ***5*** | **5** | **5** | **5** | **5** | **5** | **5** | 4 | **5** | 4 | **5** | 4 | **5** | 4 | **5** | **5** | **5** |
| Pipesworld | 100 | **29** | ***29*** | ***29*** | ***29*** | ***29*** | **29** | 28 | 28 | **29** | ***29*** | 28 | ***29*** | 28 | ***29*** | 28 | ***29*** | **29** | **29** | **29** |
| PSR | 50 | 49 | 49 | *49* | **50** | **50** | 49 | 49 | **50** | **50** | 49 | 49 | 49 | 49 | 49 | **50** | **50** | **50** | **50** | **50** |
| Rovers | 40 | 7 | 8 | 8 | 8 | 8 | **9** | **9** | **9** | 8 | 8 | **9** | 8 | **9** | **9** | **9** | 8 | **9** | **9** | **9** |
| Satellite | 36 | 7 | 7 | **12** | 9 | 8 | **12** | **12** | **12** | 9 | 8 | **12** | **12** | **12** | **12** | **12** | **12** | **12** | **12** | **12** |
| Scanalyzer | 30 | **15** | ***15*** | ***15*** | **15** | **15** | **15** | **15** | **15** | 12 | ***15*** | 11 | ***15*** | 11 | ***15*** | 11 | ***15*** | 12 | 12 | **15** |
| Tetris | 17 | **6** | ***6*** | ***6*** | 5 | ***6*** | **6** | ***6*** | 5 | 5 | ***6*** | 5 | ***6*** | 5 | ***6*** | 5 | ***6*** | **6** | **6** | **6** |
| Tidybot | 30 | 18 | *18* | *18* | 18 | *18* | 18 | *18* | 18 | 18 | 18 | 18 | **19** | 18 | **19** | 18 | 18 | **19** | **19** | **19** |
| TPP | 30 | 6 | **19** | 6 | 6 | 6 | 7 | 7 | **19** | 6 | 16 | 7 | **19** | 7 | **19** | 6 | **19** | **19** | **19** | **19** |
| Transport | 59 | **17** | ***17*** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | **17** | 12 | **17** | **17** | **17** |
| Trucks | 30 | 10 | *10* | *10* | **11** | **11** | **11** | **11** | **11** | 10 | 10 | 10 | **11** | 10 | 10 | 10 | 10 | **11** | **11** | **11** |
| Woodworking | 30 | 17 | 21 | 22 | 20 | *17* | 21 | 22 | 23 | 17 | 22 | 21 | 22 | 22 | 22 | 21 | 23 | **24** | **24** | **24** |
| Others | 478 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 | 214 |
| $\sum$ | 1630 | 790 | 821 | 800 | 803 | 802 | 813 | 810 | 837 | 772 | 821 | 792 | 836 | 794 | 832 | 800 | 823 | 837 | 840 | **850** |

Table 7: Per-domain coverage results (+Base) of the same configurations as shown in Table 3 with A$^*$ search with $h^{\text{LM-cut}}$. We use bold face to highlight highest coverage and italic font to indicate domains in which a decoupled search configuration abstained in all instances, so the number shown corresponds fully to Base.

| Domain | # | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L20 | bM | bM10s | M | M10s | F | F25s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Airport | 50 | 36 | *36* | *36* | *36* | 34 | 35 | 35 | 35 | 34 | 32 | 35 | 40 | 35 | 40 | 35 | 40 | 40 | 40 | **41** |
| Barman | 40 | 20 | *20* | *20* | 20 | *20* | 1 | 3 | 4 | 8 | *20* | 7 | *20* | 2 | *20* | 5 | *20* | 18 | 18 | **29** |
| Childsnack | 20 | 0 | *0* | 2 | 2 | *0* | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 | **3** | **3** | **3** |
| DataNetwork | 20 | **7** | ***7*** | ***7*** | 6 | 6 | 3 | 3 | **7** | 3 | 6 | 2 | 6 | 4 | 6 | 3 | 5 | **7** | **7** | **7** |
| Depots | 22 | 15 | *15* | 19 | 19 | 16 | 19 | 19 | 19 | 17 | 19 | 18 | 19 | 16 | 19 | 19 | 19 | **20** | **20** | **20** |
| Driverlog | 20 | 18 | 19 | *18* | 19 | 18 | 19 | 19 | **20** | 18 | 19 | **20** | **20** | 19 | 19 | **20** | **20** | **20** | **20** | **20** |
| Elevators | 40 | 39 | *39* | **40** | 39 | *39* | **40** | **40** | **40** | 29 | **40** | 34 | **40** | 34 | **40** | **40** | **40** | **40** | **40** | **40** |
| Floortile | 40 | 8 | *8* | 6 | 4 | 8 | 8 | 36 | 36 | 5 | 36 | 36 | 36 | 36 | 36 | 36 | 36 | **37** | **37** | **37** |
| Freecell | 80 | **80** | ***80*** | ***80*** | 79 | ***80*** | 79 | **80** | **80** | 79 | ***80*** | 79 | ***80*** | **80** | ***80*** | **80** | ***80*** | **80** | **80** | **80** |
| Grid | 5 | **4** | ***4*** | ***4*** | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | **4** | **4** | **4** |
| Hiking | 20 | 19 | *19* | *19* | *19* | *19* | **20** | **20** | **20** | 5 | **20** | 2 | **20** | 2 | **20** | **20** | **20** | **20** | **20** | **20** |
| Logistics | 63 | 52 | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | 56 | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** |
| Maintenance | 20 | 6 | *6* | *6* | 5 | 6 | 5 | 12 | 14 | 9 | 6 | 14 | 6 | 11 | 6 | 11 | 6 | **16** | **16** | **16** |
| Mprime | 35 | 31 | *31* | *31* | 31 | *31* | 29 | 28 | 29 | 27 | 31 | 30 | 31 | 28 | 31 | 28 | 31 | 31 | **32** | **32** |
| Mystery | 30 | 18 | *18* | **19** | 18 | *18* | 18 | 17 | **19** | 16 | 18 | 16 | 17 | 17 | 17 | 17 | **19** | **19** | **19** | **19** |
| NoMystery | 20 | 8 | **19** | *8* | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** |
| Openstacks | 90 | **90** | ***90*** | ***90*** | **90** | **90** | 80 | 50 | 80 | 51 | **90** | 52 | **90** | 57 | **90** | 57 | **90** | 87 | 87 | **90** |
| Organic-split | 20 | **12** | ***12*** | **12** | 11 | **12** | ***12*** | ***12*** | 11 | **12** | **12** | **12** | **12** | **12** | **12** | **12** | **12** | **12** | 11 | **12** |
| ParcPrinter | 30 | **30** | ***30*** | ***30*** | **30** | ***30*** | **30** | **30** | **30** | **30** | **30** | 29 | **30** | 29 | **30** | **30** | **30** | **30** | **30** | **30** |
| Parking | 40 | **21** | ***21*** | ***21*** | ***21*** | ***21*** | *20* | *20* | ***21*** | *20* | ***21*** | *20* | ***21*** | *20* | ***21*** | *20* | ***21*** | ***21*** | ***21*** | ***21*** |
| Pathways | 30 | 11 | 13 | *11* | 13 | 13 | 16 | 16 | 22 | 20 | 21 | 18 | 18 | 20 | 21 | 20 | 21 | **29** | **29** | **29** |
| Pipesworld | 100 | 54 | *54* | *54* | *54* | *54* | 55 | 52 | 55 | 52 | *54* | 53 | *54* | 53 | *54* | 51 | *54* | 56 | 57 | **59** |
| Rovers | 40 | 22 | 22 | **40** | 21 | 22 | 22 | 22 | **40** | **40** | 34 | 24 | 23 | 26 | 26 | 39 | **40** | **40** | **40** | **40** |
| Satellite | 36 | 25 | 26 | **36** | 26 | 26 | **36** | **36** | **36** | 31 | 33 | 30 | 31 | **36** | **36** | **36** | **36** | **36** | **36** | **36** |
| Scanalyzer | 30 | 27 | *27* | *27* | 27 | 27 | 26 | 26 | 27 | 28 | 26 | 28 | 27 | 27 | 27 | 27 | 27 | 28 | 28 | **30** |
| Storage | 30 | 19 | *19* | *19* | *19* | *19* | *19* | 20 | 21 | 20 | *19* | 21 | 19 | 19 | 19 | 22 | 19 | **23** | **23** | **23** |
| Tetris | 20 | 7 | *7* | *7* | 6 | *7* | *7* | *7* | 6 | *7* | 7 | *7* | 5 | *7* | 5 | *7* | 5 | **8** | **8** | **8** |
| Thoughtful | 20 | 8 | *8* | *8* | 8 | 8 | 8 | 8 | 8 | 9 | *8* | 8 | 10 | 9 | 10 | 8 | 9 | **13** | **13** | **13** |
| Tidybot | 20 | **16** | ***16*** | ***16*** | 15 | **16** | ***16*** | ***16*** | 15 | ***16*** | 14 | ***16*** | 15 | ***16*** | **16** | ***16*** | **16** | **16** | **16** | **16** |
| TPP | 30 | 23 | 24 | **29** | 27 | 23 | 23 | 22 | **29** | 20 | **29** | 20 | 21 | 21 | 25 | 23 | 25 | **29** | **29** | **29** |
| Transport | 58 | 17 | *17* | **58** | 17 | 17 | **58** | **58** | **58** | 13 | **58** | 14 | **58** | 14 | **58** | **58** | **58** | **58** | **58** | **58** |
| Trucks | 30 | 14 | *14* | *14* | 14 | 14 | 14 | 14 | 14 | 13 | 14 | 13 | 13 | 14 | 12 | 13 | 14 | **16** | **16** | **16** |
| Woodworking | 40 | 34 | 36 | 38 | 36 | 34 | **40** | **40** | **40** | 32 | 36 | **40** | **40** | **40** | 39 | **40** | **40** | **40** | **40** | **40** |
| Others | 497 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 | 404 |
| $\sum$ | 1686 | 1195 | 1224 | 1292 | 1221 | 1217 | 1247 | 1250 | 1327 | 1152 | 1317 | 1187 | 1313 | 1193 | 1326 | 1282 | 1344 | 1383 | 1384 | 1404 |

Table 8: Per-domain coverage results (+Base) of the same configurations as shown in Table 4 with GBFS using $h^{\mathrm{FF}}$. We use bold face to highlight highest coverage and italic font to indicate domains in which a decoupled search configuration abstained in all instances, so the number shown corresponds fully to Base.

| Domain | # | Base | Fo | IF | IA | MIS | LPS | LPG | O(o) | L | L20 | bM | bM30s | M | M5 | F | F20s | O(n) | O(d) | O(a) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Agricola | 20 | **12** | ***12*** | ***12*** | **12** | ***12*** | **12** | **12** | **12** | **12** | ***12*** | 11 | ***12*** | 11 | **12** | **12** | ***12*** | **12** | **12** | **12** |
| Airport | 50 | 37 | *37* | *37* | *37* | 36 | 36 | 36 | 36 | 34 | 34 | 35 | 34 | 35 | 35 | 36 | 40 | 40 | **41** | **41** |
| Barman | 40 | 28 | *28* | *28* | 28 | *28* | 22 | 19 | 29 | 26 | *28* | 27 | *28* | 25 | 27 | 24 | *28* | **40** | **40** | **40** |
| Childsnack | 20 | 6 | *6* | **20** | **20** | *6* | 0 | 0 | **20** | 2 | **20** | 3 | **20** | 5 | **20** | 4 | **20** | **20** | **20** | **20** |
| DataNetwork | 20 | 10 | *10* | *10* | 7 | 9 | 10 | 8 | 10 | 6 | 7 | 8 | 7 | 9 | 9 | 6 | 7 | 10 | 10 | **11** |
| Depots | 22 | 19 | *19* | **21** | **21** | 19 | **21** | **21** | **21** | **21** | **21** | 20 | **21** | 20 | **21** | **21** | **21** | **21** | **21** | **21** |
| Driverlog | 20 | **20** | **20** | ***20*** | **20** | **20** | 19 | 19 | **20** | **20** | **20** | **20** | **20** | 19 | 19 | **20** | **20** | **20** | **20** | **20** |
| Elevators | 40 | 39 | *39* | **40** | 39 | *39* | **40** | **40** | **40** | 39 | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** |
| Floortile | 40 | 8 | *8* | 9 | 4 | 8 | 9 | **39** | **39** | 5 | **39** | **39** | 38 | **39** | **39** | **39** | **39** | **39** | **39** | **39** |
| Freecell | 80 | **80** | ***80*** | ***80*** | **80** | ***80*** | **80** | **80** | **80** | 78 | ***80*** | 79 | ***80*** | **80** | ***80*** | **80** | ***80*** | **80** | **80** | **80** |
| Grid | 5 | 4 | *4* | *4* | **5** | **5** | 4 | 4 | **5** | 4 | **5** | 4 | **5** | **5** | **5** | 4 | **5** | **5** | **5** | **5** |
| Hiking | 20 | **20** | ***20*** | ***20*** | ***20*** | ***20*** | **20** | **20** | **20** | 17 | **20** | 12 | **20** | 17 | **20** | **20** | **20** | **20** | **20** | **20** |
| Logistics | 63 | 61 | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** | **63** |
| Maintenance | 20 | 9 | *9* | *9* | 9 | 9 | 10 | 14 | 16 | 13 | 8 | 13 | 8 | 16 | 14 | **17** | 9 | **17** | **17** | **17** |
| Mystery | 30 | 16 | *16* | 16 | 16 | *16* | 16 | 17 | 17 | 17 | 16 | 16 | 16 | 17 | **18** | 17 | 15 | **18** | **18** | **18** |
| NoMystery | 20 | 9 | **19** | *9* | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** | **19** |
| Openstacks | 90 | **90** | ***90*** | ***90*** | **90** | **90** | 68 | 17 | 68 | 13 | **90** | 20 | **90** | 34 | 84 | 33 | **90** | 86 | 86 | **90** |
| Pathways | 30 | 21 | 21 | *21* | 21 | 22 | 19 | 18 | 25 | 21 | 25 | 21 | 22 | 22 | 25 | 22 | 24 | **29** | **29** | **29** |
| Pipesworld | 100 | 82 | *82* | *82* | *82* | *82* | 82 | 83 | 83 | **84** | *82* | **84** | *82* | **84** | *82* | **84** | *82* | **84** | **84** | **84** |
| Satellite | 36 | **36** | **36** | **36** | 34 | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** | **36** |
| Scanalyzer | 30 | 28 | *28* | *28* | 28 | 28 | 28 | 28 | 28 | **30** | 28 | **30** | *28* | **30** | 28 | **30** | 28 | **30** | **30** | **30** |
| Spider | 20 | 13 | *13* | *13* | *13* | **14** | *13* | *13* | **14** | *13* | *13* | *13* | *13* | *13* | *13* | *13* | *13* | *13* | **14** | **14** |
| Storage | 30 | 20 | *20* | *20* | *20* | *20* | *20* | 21 | 21 | 20 | *20* | 20 | *20* | 20 | 20 | **22** | 20 | **22** | **22** | **22** |
| Tetris | 20 | 14 | *14* | *14* | **15** | *13* | *13* | *13* | **15** | *13* | 13 | *13* | *14* | *13* | 10 | *13* | 10 | **15** | **15** | **15** |
| Thoughtful | 20 | 11 | *11* | *11* | 10 | 12 | 11 | 11 | 10 | 10 | *11* | 12 | *11* | 12 | 10 | 12 | 13 | **15** | **15** | **15** |
| Tidybot | 20 | **16** | ***16*** | ***16*** | 15 | **16** | ***16*** | ***16*** | 15 | ***16*** | 13 | ***16*** | 13 | ***16*** | 13 | ***16*** | 13 | 15 | **16** | **16** |
| TPP | 30 | **30** | **30** | **30** | 26 | **30** | 28 | **30** | **30** | **30** | **30** | 28 | **30** | 29 | 27 | 28 | 25 | **30** | **30** | **30** |
| Transport | 58 | 41 | *41* | **58** | 41 | 41 | **58** | **58** | **58** | 33 | **58** | 32 | **58** | 32 | 57 | **58** | **58** | **58** | **58** | **58** |
| Trucks | 30 | 16 | *16* | *16* | 14 | 16 | 14 | 14 | 16 | 14 | 14 | 14 | 17 | 16 | 17 | 15 | 16 | 19 | 19 | **20** |
| Woodworking | 40 | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** | 34 | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** | **40** |
| Others | 622 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 | 534 |
| $\sum$ | 1686 | 1370 | 1382 | 1407 | 1383 | 1383 | 1361 | 1343 | 1440 | 1277 | 1439 | 1322 | 1439 | 1351 | 1437 | 1378 | 1440 | 1490 | 1493 | 1499 |

Table 9: Per-domain coverage results (+Base) of the same configurations as shown in Table 5 with GBFS using $h^{\mathrm{FF}}$ and PO. We use bold face to highlight highest coverage and italic font to indicate domains in which a decoupled search configuration abstained in all instances, so the number shown corresponds fully to Base.

| Domain | # | bM | bMO | bMs | bMOs |
|---|---|---|---|---|---|
| Airport | 41 | 443778 | 519862 | 443778 | 519862 |
| Childsnack | 22 | 489075 | 596215 | 489075 | 596215 |
| DataNetwork | 39 | 3765 | 114159 | 3765 | 114159 |
| Depots | 15 | 177054 | 386855 | 177054 | 386855 |
| Driverlog | 9 | 527 | 2888 | 540 | 19283 |
| Elevators | 61 | 2810 | 3033 | 2822 | 47613 |
| Floortile | 80 | 5798 | 71122 | 5798 | 71122 |
| Freecell | 80 | 31312 | 32853 | 31312 | 32853 |
| GED | 39 | 36757 | 478262 | 36757 | 478262 |
| Grid | 4 | 1039 | 4336 | 1039 | 4336 |
| Gripper | 15 | 1266 | 1705 | 1269 | 14815 |
| Hiking | 40 | 9582 | 20684 | 9582 | 20684 |
| Logistics | 37 | 173 | 73130 | 173 | 73130 |
| Maintenance | 24 | 10498 | 11294 | 10498 | 11294 |
| Miconic | 145 | 247 | 409 | 247 | 409 |
| Mprime | 24 | 8894 | 9584 | 8903 | 49035 |
| Mystery | 16 | 3813 | 4198 | 3831 | 57335 |
| NoMystery | 40 | 62 | 82 | 62 | 82 |
| Openstacks | 110 | 8700 | 10519 | 8700 | 10519 |
| Organic | 1 | 129 | 305 | 129 | 305 |
| ParcPrinter | 1 | 342 | 12764 | 436 | 43566 |
| Parking | 13 | 945503 | 965463 | 945503 | 965463 |
| Pathways | 11 | 4264 | 34570 | 4294 | 57181 |
| Pipesworld | 51 | 287124 | 295369 | 287124 | 295369 |
| PSR | 27 | 227 | 40653 | 231 | 73424 |
| Rovers | 24 | 2830 | 5912 | 2941 | 101162 |
| Satellite | 29 | 1216 | 194623 | 1328 | 199237 |
| Scanalyzer | 21 | 9969 | 11301 | 9969 | 11301 |
| Storage | 20 | 212972 | 219786 | 212972 | 219786 |
| Tetris | 6 | 756398 | 771894 | 756398 | 771894 |
| TPP | 12 | 1827 | 11028 | 1937 | 71156 |
| Transport | 117 | 692 | 784 | 700 | 8911 |
| Trucks | 16 | 1257 | 18545 | 1233 | 70082 |
| Woodworking | 11 | 2901 | 257168 | 2953 | 387977 |
| Zenotravel | 20 | 172 | 847 | 172 | 847 |

Table 10: Comparison of the average ILP encoding size of some variants of the balanced leaf mobility optimization function on instances where all methods are able to fully construct the ILP.