From Fork Decoupling to Star-Topology Decoupling

Daniel Gnad and **Jörg Hoffmann**
Saarland University
Saarbrücken, Germany
{gnad, hoffmann}@cs.uni-saarland.de

Carmel Domshlak
Technion
Haifa, Israel
dcarmel@ie.technion.ac.il

Abstract

Fork decoupling is a recent approach to exploiting problem structure in state space search. The problem is assumed to take the form of a fork, where a single (large) *center* component provides preconditions for several (small) *leaf* components. The leaves are then conditionally independent in the sense that, given a fixed center path π^C, the *compliant* leaf moves – those leaf moves enabled by the preconditions supplied along π^C – can be scheduled independently for each leaf. *Fork-decoupled state space search* exploits this through conducting a regular search over center paths, augmented with maintenance of the compliant paths for each leaf individually. We herein show that the same ideas apply to much more general *star-topology* structures, where leaves may supply preconditions for the center, and actions may affect several leaves simultaneously as long as they also affect the center. Our empirical evaluation in planning, super-imposing star topologies by automatically grouping the state variables into suitable components, shows the merits of the approach.

Introduction

In classical AI planning, large deterministic transition systems are described in terms of a set of finite-domain state variables, and actions specified via preconditions and effects over these state variables. The task is to find a sequence of actions leading from a given initial state to a state that satisfies a given goal condition. *Factored planning* is one traditional approach towards doing so effectively. The idea is to decouple the planning task into subsets, *factors*, of state variables. In *localized* factored planning (Amir and Engelhardt 2003; Brafman and Domshlak 2006; 2008; 2013; Fabre et al. 2010), two factors interact if they are affected by common actions, and a global plan needs to comply with these *cross-factor interactions*. In *hierarchical* factored planning (e. g. (Knoblock 1994; Kelareva et al. 2007; Wang and Williams 2015)), the factors are used within a hierarchy of increasingly more detailed abstraction levels, search refining abstract plans as it proceeds down the hierarchy, and backtracking if an abstract plan has no refinement.

Both localized and hierarchical factored planning have traditionally been viewed as the resolution of complex interactions between (relatively) small factors. In recent work, Gnad and Hoffmann (2015) (henceforth: GH) proposed to turn this upside down, fixing a *simple* interaction profile the factoring should induce, at the cost of potentially very *large* factors. They baptize this approach *target-profile factoring*, and develop a concrete instance where the target profile is a *fork*: a single (large) *center* factor provides preconditions for several (small) *leaf* factors, and no other cross-factor interactions are present. They introduce a simple and quick *factoring strategy* which, given an arbitrary input planning task, analyzes the state variable dependencies and either outputs a *fork factoring*, or *abstains* if the relevant stucture is not there (the task can then be handed over to other methods).

Say the factoring strategy does not abstain. We then face, not a general planning problem, but a fork planning problem. GH's key observation is that these can be solved via *fork-decoupled state space search*: The leaves are conditionally independent in the sense that, given a fixed center path π^C, the *compliant* leaf moves – those leaf moves enabled by the preconditions supplied along π^C – can be scheduled independently for each leaf. This can be exploited by searching only over center paths, and maintaining the compliant paths separately for each leaf, thus avoiding the enumeration of state combinations across leaves. GH show that this substantially reduces the number of reachable states in (non-abstained) planning benchmarks, almost always by at least $1 - 2$ orders of magnitude, up to 6 orders of magnitude in one domain (TPP). They show how to connect to classical planning heuristics, and to standard search methods, guaranteeing plan optimality under the same conditions as before.

We herein extend GH's ideas to *star-topology* structures, where the center still supplies preconditions for the leaves, but also the leaves may supply preconditions for the center, and actions may affect several leaves simultaneously as long as they also affect the center. The connection to standard heuristics and search methods remains valid. We run experiments on the planning competition benchmarks.

We place our work in the planning context for the direct connection to GH. However, note that trying to factorize a general input problem, and having to abstain in case the saught structure is not present, really is an artefact of the general-planning context. Star-topology decoupling applies, in principle, to the state space of any system that naturally has a star topology. As star topology is a classical design paradigm in many areas of CS, such systems abound.

Preliminaries

We use a finite-domain state variable formalization of planning (e. g. (Bäckström and Nebel 1995; Helmert 2006)). A *finite-domain representation* planning task, short FDR task, is a quadruple $\Pi = \langle V, A, I, G \rangle$. V is a set of *state variables*, where each $v \in V$ is associated with a finite domain $\mathcal{D}(v)$. We identify (partial) variable assignments with sets of variable/value pairs. A complete assignment to V is a *state*. I is the *initial state*, and the *goal* G is a partial assignment to V. A is a finite set of *actions*. Each action $a \in A$ is a triple $\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle$ where the *precondition* $\mathsf{pre}(a)$ and *effect* $\mathsf{eff}(a)$ are partial assignments to V, and $\mathsf{cost}(a) \in \mathbb{R}^{0+}$ is the action's non-negative *cost*.

For a partial assignment p, $\mathcal{V}(p) \subseteq V$ denotes the subset of state variables instantiated by p. For any $V' \subseteq \mathcal{V}(p)$, by $p[V']$ we denote the assignment to V' made by p. An action a is *applicable* in a state s if $\mathsf{pre}(a) \subseteq s$, i. e., if $s[v] = \mathsf{pre}(a)[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a in s changes the value of each $v \in \mathcal{V}(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$, and leaves s unchanged elsewhere; the outcome state is denoted $s[\![a]\!]$. We also use this notation for partial states p: by $p[\![a]\!]$ we denote the assignment over-writing p with $\mathsf{eff}(a)$ where both p and $\mathsf{eff}(a)$ are defined. The outcome state of applying a sequence of (respectively applicable) actions is denoted $s[\![\langle a_1, \dots, a_n \rangle]\!]$. A *plan* for Π is an action sequence s.t. $G \subseteq I[\![\langle a_1, \dots, a_n \rangle]\!]$. The plan is *optimal* if its summed-up cost is minimal among all plans for Π.

The *causal graph* of a planning task captures state variable dependencies (e. g. (Knoblock 1994; Jonsson and Bäckström 1995; Brafman and Domshlak 2003; Helmert 2006)). We use the commonly employed definition in the FDR context, where the causal graph *CG* is a directed graph over vertices V, with an arc from v to v', which we denote $(v \rightarrow v')$, if $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in [\mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a))] \times \mathcal{V}(\mathsf{eff}(a))$. In words, the causal graph captures precondition-effect as well as effect-effect dependencies, as result from the action descriptions. A simple intuition is that, whenever $(v \rightarrow v')$ is an arc in *CG*, changing the value of v' may involve changing that of v as well. We assume for simplicity that *CG* is weakly connected (this is wlog: else, the task can be equivalently split into several independent tasks).

We will also need the notion of a *support graph*, *SuppG*, similarly as used e. g. by Hoffmann (2011). *SuppG* is like *CG* except its arcs are only those $(v \rightarrow v')$ where there exists an action $a \in A$ such that $(v, v') \in \mathcal{V}(\mathsf{pre}(a)) \times \mathcal{V}(\mathsf{eff}(a))$. In words, the support graph captures only the precondition-effect dependencies, not effect-effect dependencies. This more restricted concept will be needed to conveniently describe our notion of star topologies, for which purpose the effect-effect arcs in *CG* are not suitable.

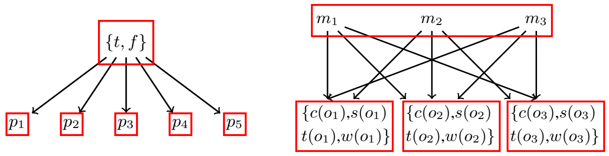
$\{t, f\}$
p_1
p_2
p_3
p_4
p_5
m_1
m_2
m_3
$\{c(o_1), s(o_1)$ $t(o_1), w(o_1)\}$
$\{c(o_2), s(o_2)$ $t(o_2), w(o_2)\}$
$\{c(o_3), s(o_3)$ $t(o_3), w(o_3)\}$

Figure 1: (Gnad and Hoffmann 2015) Possible fork factorings in transportation with fuel consumption (left), and job-planning problems (right).

As an illustrative example, we will consider a simple transportation-like FDR planning task $\Pi = \langle V, A, I, G \rangle$ with one package p and two trucks t_A, t_B, defined as follows. $V = \{p, t_A, t_B\}$ where $\mathcal{D}(p) = \{A, B, l_1, l_2, l_3\}$ and $\mathcal{D}(t_A) = \mathcal{D}(t_B) = \{l_1, l_2, l_3\}$. The initial state is $I = \{p = l_1, t_A = l_1, t_B = l_3\}$, i. e., p and t_A start at l_1, and t_B starts at l_3. The goal is $G = \{p = l_3\}$. The actions (all with cost 1) are truck moves and load/unload:

- move(x, y, z): precondition $\{t_x = y\}$ and effect $\{t_x = z\}$, where $x \in \{A, B\}$ and $\{y, z\} \in \{\{l_1, l_2\}, \{l_2, l_3\}\}$.
- load(x, y): precondition $\{t_x = y, p = y\}$ and effect $\{p = x\}$, where $x \in \{A, B\}$ and $y \in \{l_1, l_2, l_3\}$.
- unload(x, y): precondition $\{t_x = y, p = x\}$ and effect $\{p = y\}$, where $x \in \{A, B\}$ and $y \in \{l_1, l_2, l_3\}$.

The causal graph and support graph of this task are identical. Their arcs are $(t_A \rightarrow p)$ and $(t_B \rightarrow p)$.

Fork Decoupling

We give a brief summary of fork decoupling, in a form suitable for describing our extension to star topologies.

Definition 1 (Fork Factoring (GH)) *Let Π be an FDR task with variables V. A* factoring *$\mathcal{F}$ is a partition of V into non-empty subsets F, called* factors*. The* interaction graph *$IG(\mathcal{F})$ of $\mathcal{F}$ is the directed graph whose vertices are the factors, with an arc $(F \rightarrow F')$ if $F \neq F'$ and there exist $v \in F$ and $v' \in F'$ such that $(v \rightarrow v')$ is an arc in CG.*

$\mathcal{F}$ is a fork factoring *if $|\mathcal{F}| > 1$ and there exists $F^C \in \mathcal{F}$ s.t. the arcs in $IG(\mathcal{F})$ are exactly $\{(F^C \rightarrow F^L) \mid F^L \in \mathcal{F} \setminus \{F^C\}\}$. F^C is the* center *of $\mathcal{F}$, and all other factors $F^L \in \mathcal{F}^L := \mathcal{F} \setminus \{F^C\}$ are* leaves*. We also consider the* trivial factoring *where $F^C = V$ and $\mathcal{F}^L = \emptyset$, and the* pathological factoring *where $F^C = \emptyset$ and $\mathcal{F}^L = \{V\}$.*

The only cross-factor interactions in a fork factoring consist in the center factor establishing preconditions for actions moving individual leaf factors. We use the word "center", instead of GH's "root", to align the terminology with star topologies. Regarding the trivial and pathological cases, in the former decoupled search simplifies to standard search, and in the latter the entire problem is pushed into the single "leaf". Note that, when we say that $\mathcal{F}$ is a fork factoring, we explicitly exclude these cases.

In our example, we will consider the fork factoring where $F^C = \{t_A, t_B\}$ and the single leaf is $F^L = \{p\}$.

Given an arbitrary FDR task Π as input, as pointed out by GH, a fork factoring – if one exists, which is the case iff the causal graph has more than one strongly connected component (SCC) – can be found automatically based on a simple causal graph analysis. We describe GH's strategy later, in the experiments, along with our own generalized strategies. For illustration, Figure 1 already shows factorings that GH's strategy may find, on practical problems akin to planning benchmarks. On the left, a truck t with fuel supply f transports packages $p_1, \dots, p_n$. On the right, objects o_i are independent except for sharing the machines.

We need some terminology, that we will use also for star topologies later on. Assume an FDR task $\Pi = \langle V, A, I, G \rangle$ and a fork factoring $\mathcal{F}$ with center F^C and leaves $F^L \in \mathcal{F}^L$. We refer to the actions A^C affecting the center as *center actions*, notation convention a^C, and to all other actions as *leaf actions*, notation convention a^L. For the set of actions affecting one particular $F^L \in \mathcal{F}^L$, we write $A^L|_{F^L}$. As $\mathcal{F}$ is a fork factoring, the center actions A^C have preconditions and effects only on F^C. The leaf actions $A^L|_{F^L}$ have preconditions only on $F^C \cup F^L$, and effects only on F^L. The sets A^C and $A^L|_{F^L}$ form a partition of the original action set A.

A *center path* is a sequence of center actions applicable to I; a *leaf path* is a sequence of leaf actions applicable to I when ignoring preconditions on the center. Value assignments to F^C are *center states*, notated s^C, and value assignments to any $F^L \in \mathcal{F}^L$ are *leaf states*, notated s^L. For the leaf states of one particular $F^L \in \mathcal{F}^L$, we write $S^L|_{F^L}$, and for the set of all leaf states we write S^L. A center state s^C is a *goal center state* if $s^C \supseteq G[F^C]$, and a leaf state $s^L \in S^L|_{F^L}$ is a *goal leaf state* if $s^L \supseteq G[F^L]$.

The idea in fork-decoupling is to augment a regular search over center paths with maintenance of cheapest compliant leaf paths for each leaf. A leaf path $\pi^L = \langle a^L_1, \dots, a^L_n \rangle$ complies with center path π^C if we can schedule the a^L_i at monotonically increasing points alongside π^C so that each a^L_i is enabled in the respective center state. Formally:

Definition 2 (Fork-Compliant Path (GH)) *Let* Π *be an FDR task,* $\mathcal{F}$ *a fork factoring with center* F^C*, and* π^C *a center path traversing center states* $\langle s^C_0, \dots, s^C_n \rangle$*. For a leaf path* $\pi^L = \langle a^L_1, \dots, a^L_m \rangle$*, an* embedding *into* π^C *is a monotonically increasing function* $t : \{1, \dots, m\} \mapsto \{0, \dots, n\}$ *so that, for every* $i \in \{1, \dots, m\}$, $\mathsf{pre}(a^L_i)[F^C] \subseteq s^C_{t(i)}$*. We say that* π^L fork-complies *with* π^C*, also* π^L *is* π^C-fork-compliant*, if an embedding exists.*

Where the center path in question is clear from context, or when discussing compliant paths in general, we will omit "π^C" and simply talk about *compliant* leaf paths.

In our example, $\pi^L = \langle \text{load}(A, l_1) \rangle$ complies with the empty center path, and $\pi^L = \langle \text{load}(A, l_1), \text{unload}(A, l_2) \rangle$ complies with the center path $\pi^C = \langle \text{move}(A, l_1, l_2) \rangle$. But $\pi^L = \langle \text{load}(A, l_1), \text{unload}(A, l_3) \rangle$ does not comply with π^C as the required precondition $t_A = l_3$ is not established on π^C. And $\pi^L = \langle \text{load}(A, l_1)$, $\text{unload}(A, l_2)$, $\text{load}(A, l_2)$, $\text{unload}(A, l_1) \rangle$ does not comply with π^C as the last precondition $t_A = l_1$ does not appear behind $t_A = l_2$ on π^C.

The notion of compliant paths is a reformulation of plans for the original input planning task Π, in the following sense. Say π is a plan for Π, and say π^C is the sub-sequence of center actions in π. Then π^C is a center path. For each leaf $F^L \in \mathcal{F}^L$, say π^L is the sub-sequence of $A^L|_{F^L}$ actions in π. Then π^L is a leaf path, and is π^C-fork-compliant because π schedules π^L along with π^C in a way so that its center preconditions are fulfilled. Vice versa, if a center path π^C reaches a goal center state, and can be augmented with π^C-fork-compliant leaf paths π^L reaching goal leaf states, then the embedding of the π^L into π^C yields a plan for Π. Hence *the plans for* Π *are in one-to-one correspondence with center paths augmented with compliant leaf paths.*

GH define the *fork-decoupled state space* Θ^ϕ, in which each *fork-decoupled state* s is a pair $\langle \mathsf{center}(s), \mathsf{prices}(s) \rangle$ of a center state $\mathsf{center}(s)$ along with a *pricing function* $\mathsf{prices}(s) : S^L \mapsto \mathbb{R}^{0+} \cup \{\infty\}$. The paths in Θ^ϕ correspond to center paths, i. e., the *fork-decoupled initial state* I^ϕ has $\mathsf{center}(I^\phi) = I[F^C]$, and the transitions over center states are exactly those induced by the center actions. The pricing functions are maintained so that, for every center path π^C ending in fork-decoupled state s, and for every leaf state s^L, $\mathsf{prices}(s)[s^L]$ equals the cost of a cheapest π^C-fork-compliant leaf path π^L ending in s^L. The *fork-decoupled goal states* are those s where $\mathsf{center}(s)$ is a goal center state, and, for every $F^L \in \mathcal{F}^L$, at least one goal leaf state $s^L \in S^L|_{F^L}$ has a finite price $\mathsf{prices}(s)[s^L] < \infty$. Once a fork-decoupled goal state s is reached, a plan for Π can be extracted by augmenting the center path π^C leading to s with cheapest π^C-fork-compliant *goal leaf paths*, i. e., leaf paths ending in goal leaf states. Observe that this plan is optimal subject to fixing π^C, i. e., the cheapest possible plan for Π when comitting to exactly the center moves π^C.

Say $\pi^C = \langle \text{move}(A, l_1, l_2), \text{move}(B, l_3, l_2), \text{move}(B, l_2, l_3) \rangle$ in our example, traversing the fork-decoupled states s_0, s_1, s_2, s_3. Then $\mathsf{prices}(s_0)[p = l_1] = 0$, $\mathsf{prices}(s_0)[p = A] = 1$, $\mathsf{prices}(s_1)[p = l_2] = 2$, $\mathsf{prices}(s_2)[p = B] = 3$, and $\mathsf{prices}(s_3)[p = l_3] = 4$. To extract a plan for Π from the fork-decoupled goal state s_3, we trace back the compliant leaf path supporting $p = l_3$ and embed it into π^C. The resulting plan loads p onto t_A, moves t_A to l_2, unloads p, moves t_B to l_2, loads p onto t_2, moves t_B to l_3, and unloads p.

The core of GH's construction is the maintenance of pricing functions. For forks, this is simple enough to be described in a few lines within the definition of Θ^ϕ. For star topologies, we need to substantially extend this construction, so we hone in on it in more detail here. We reformulate it in terms of *compliant path graphs*, which capture all possible compliant graphs for a leaf F^L given a center path π^C:

Definition 3 (Fork-Compliant Path Graph) *Let* Π *be an FDR task,* $\mathcal{F}$ *a fork factoring with center* F^C *and leaves* $\mathcal{F}^L$*, and* π^C *a center path traversing center states* $\langle s^C_0, \dots, s^C_n \rangle$*. The* π^C-fork-compliant path graph *for a leaf* $F^L \in \mathcal{F}^L$*, denoted* $\mathsf{CompG}^\phi(\pi^C, F^L)$*, is the arc-labeled weighted directed graph whose vertices are the* time-stamped leaf states $\{s^L_t \mid s^L \in S^L|_{F^L}, 0 \leq t \leq n\}$*, and whose arcs are:*

(i) $s^L_t \xrightarrow{a^L} {s'}^L_t$ *with weight* $c(a^L)$ *whenever* $s^L, {s'}^L \in S^L|_{F^L}$ *and* $a^L \in A^L|_{F^L}$ *such that* $\mathsf{pre}(a^L)[F^C] \subseteq s^C_t$, $\mathsf{pre}(a^L)[F^L] \subseteq s^L$*, and* $s^L[\![a^L]\!] = {s'}^L$*.*

(ii) $s^L_t \xrightarrow{0} s^L_{t+1}$ *with weight* 0 *for all* $s^L \in S^L|_{F^L}$ *and* $0 \leq t < n$*.*

In words, the π^C-fork-compliant path graph includes a copy of the leaf states at every time step $0 \leq t \leq n$ along the center path π^C. Within each t, the graph includes all leaf-state transitions enabled in the respective center state. From each t to $t + 1$, the graph has a 0-cost transition for each leaf state. Consider again the example, and the center path $\pi^C = \langle \text{move}(A, l_1, l_2)\rangle$. The π^C-fork-compliant path graph for the package is shown in Figure 2.[1]

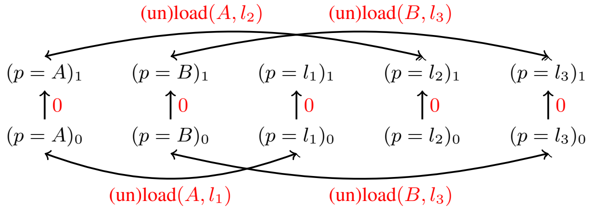

Figure 2: The fork-compliant path graph for $\pi^C = \langle \text{move}(A, l_1, l_2)\rangle$ in our illustrative example.

The π^C-fork-compliant leaf path $\pi^L = \langle \text{load}(A, l_1), \text{unload}(A, l_2)\rangle$ can be embedded by starting at $(p = l_1)_0$, following the arc labeled $\text{load}(A, l_1)$ to $(p = A)_0$, following the 0-arc to $(p = A)_1$, and following the arc labeled $\text{unload}(A, l_2)$ to $(p = l_2)_1$. The non-compliant leaf path $\pi^L = \langle \text{load}(A, l_1), \text{unload}(A, l_2), \text{load}(A, l_2), \text{load}(A, l_1)\rangle$ cannot be embedded, as (un)load(A, l_2) appears only at $t = 1$, while $\text{load}(A, l_1)$ is not available anymore at $t \geq 1$.

Lemma 1 *Let Π be an FDR task, $\mathcal{F}$ a fork factoring with center F^C and leaves $\mathcal{F}^L$, and π^C a center path. Let $F^L \in \mathcal{F}^L$, and $s^L \in S^L|_{F^L}$. Then the cost of a cheapest π^C-fork-compliant leaf path π^L ending in s^L equals the cost of a cheapest path from $I[F^L]_0$ to s^L_n in CompG$^\phi(\pi^C, F^L)$.*

Proof: Given a π^C-fork-compliant leaf path $\pi^L = \langle a^L_1, \dots, a^L_m\rangle$, we can schedule each a^L_i as a (i) arc in *CompG*$^\phi(\pi^C, F^L)$ at the time step $t(i)$ assigned by the embedding. Connecting the resulting partial paths across time steps using the (ii) arcs, we get a path π from $I[F^L]_0$ to s^L_n in *CompG*$^\phi(\pi^C, F^L)$, whose cost equals that of π^L. Vice versa, given a path π from $I[F^L]_0$ to s^L_n in *CompG*$^\phi(\pi^C, F^L)$, remove the time indices of the vertices on π, and remove the arcs crossing time steps. This yields a π^C-fork-compliant leaf path π^L ending in s^L, whose cost equals that of π. So the π^C-fork-compliant leaf paths ending in s^L are in one-to-one correspondence with the paths from $I[F^L]_0$ to s^L_n in *CompG*$^\phi(\pi^C, F^L)$, showing the claim. □

To prepare our extension in the next section, we now reformulate the fork-decoupled state space Θ^ϕ. Each fork-decoupled state s is a center path $\pi^C(s)$, associated for every leaf $F^L \in \mathcal{F}^L$ with the π^C-fork-compliant path graph *CompG*$^\phi(\pi^C(s), F^L)$. The fork-decoupled initial state I^ϕ is the empty center path $\pi^C(I^\phi) = \langle\rangle$. The successor states s' of s are exactly the center paths extending $\pi^C(s)$ by one more center action. The fork-decoupled goal states are those s where $\pi^C(s)$ ends in a center goal state, and for every $F^L \in \mathcal{F}^L$ there exists a goal leaf state $s^L \in S^L|_{F^L}$ s.t. $s^L_{|\pi^C(s)|}$ is reachable from $I[F^L]_0$ in *CompG*$^\phi(\pi^C(s), F^L)$.

This formulation is different from GH's, but is equivalent given Lemma 1: Instead of maintaining the pricing functions prices(s) explicitly listing the costs of cheapest π^C-fork-compliant paths, we maintain the fork-compliant path graphs *CompG*$^\phi(\pi^C(s), F^L)$, from which these same costs can be obtained in terms of standard graph distance.

Star-Topology Decoupling

We now extend the concepts of compliant paths, and compliant path graphs, to handle star topologies instead of forks. A star topology is one where the center may interact arbitrarily with each leaf, and even effect-effect dependencies across leaves are allowed so long as they also affect the center:

Definition 4 (Star Factoring) *Let Π be an FDR task, and let $\mathcal{F}$ be a factoring. The* support-interaction graph *SuppIG$(\mathcal{F})$ of $\mathcal{F}$ is the directed graph whose vertices are the factors, with an arc $(F \to F')$ if $F \neq F'$ and there exist $v \in F$ and $v' \in F'$ such that $(v \to v')$ is an arc in SuppG. $\mathcal{F}$ is a* star factoring *if $|\mathcal{F}| > 1$ and there exists $F^C \in \mathcal{F}$ s.t. the following two conditions hold:*

(1) The arcs in SuppIG$(\mathcal{F})$ are contained in $\{(F^C \to F^L), (F^L \to F^C) \mid F^L \in \mathcal{F} \setminus \{F^C\}\}$.

(2) For every action a, if there exist $F^L_1, F^L_2 \in \mathcal{F} \setminus \{F^C\}$ such that $F^L_1 \neq F^L_2$ and $\mathcal{V}(\mathsf{eff}(a)) \cap F^L_1 \neq \emptyset$ as well as $\mathcal{V}(\mathsf{eff}(a)) \cap F^L_2 \neq \emptyset$, then $\mathcal{V}(\mathsf{eff}(a)) \cap F^C \neq \emptyset$.

F^C is the center *of $\mathcal{F}$, and all other factors $F^L \in \mathcal{F}^L := \mathcal{F} \setminus \{F^C\}$ are* leaves.

A star factoring $\mathcal{F}$ is strict *if the arcs in IG$(\mathcal{F})$ are contained in $\{(F^C \to F^L), (F^L \to F^C) \mid F^L \in \mathcal{F} \setminus \{F^C\}\}$.*

We cannot characterize star factorings in terms of just the causal graph, because the effect-effect arcs in that graph are inserted for all variable pairs in the effect: If there is an arc between two leaves, we cannot distinguish whether or not the same action also affects the center. In contrast, in a strict star factoring every action affects at most one leaf, which *can* be characterized in terms of just the causal graph. Hence strict star factorings are interesting from a practical perspective, allowing different/simpler factoring strategies.

Obviously, Definition 4 generalizes Definition 1. Perhaps less obvious is how far this generalization carries. As pointed out, an FDR task has a fork factoring iff its causal graph has more than one SCC. In contrast, *every* FDR task has a star factoring. In fact, any partition of the variables into two non-empty subsets is a star factoring: Calling one half of the variables the “center”, and the other the “leaf”, we have a (strict) star factoring, as Definition 4 does not apply any restrictions if there is a single leaf only.[2] That said, it

[1] Note that *CompG*$^\phi(\pi^C, F^L)$ contains redundant parts, not reachable from the initial leaf state $I[F^L]$, i. e., $(p = l_1)_0$ in the figure. This is just to keep the definition simple, in practice one can maintain only the reachable part of *CompG*$^\phi(\pi^C, F^L)$.

[2] Observe also that Definition 4 (2) can always be enforced wlog, simply by introducing redundant effects on F^C. However, actions affecting F^C are those our search must branch over, so this is just another way of saying “cross-leaf effects can be tackled by centrally branching over the respective actions”. Doing so weakens the decoupling obtained, and for now we do not consider it.

is not clear whether single-leaf factorings are useful in practice. We get back to this when discussing our experiments.

To illustrate, consider what we will refer to as the *no-empty example*, where we forbid "empty truck moves". This is as before, except the precondition of move(x, y, z) is no longer $\{t_x = y\}$, but is $\{t_x = y, p = x\}$: A truck can only move if the package is currently inside it. The causal graph arcs now are not only $(t_A \rightarrow p)$ and $(t_B \rightarrow p)$ as before, but also $(p \rightarrow t_A)$ and $(p \rightarrow t_B)$. Hence there exists no fork factoring. But our previous factoring with $F^C = \{t_A, t_B\}$ and the single leaf $F^L = \{p\}$ is now a strict star factoring.

To define compliance, we will be using the same terminology as before, i. e. center actions/paths and leaf actions/paths. These concepts are (mostly) defined as before, however their behavior is more complicated now.

The notions of center/leaf states, and of goal center/leaf states, remain the same. The center actions A^C are still all those actions affecting the center, and the leaf actions $A^L|_{F^L}$ for $F^L \in \mathcal{F}^L$ are still all those actions affecting F^L. However, *A^C and $A^L|_{F^L}$ are no longer disjoint*, as the same action may affect both A^C and $A^L|_{F^L}$.

A leaf path still is a sequence of leaf actions applicable to I when ignoring all center preconditions. The notion of center path changes, as now there may be leaf preconditions; we ignore these, i. e., *a center path is now a sequence of center actions applicable to I when ignoring all leaf preconditions*. As center and leaf paths may overlap, we need to clarify where to account for the cost of the shared actions. Our search, as we explain in a moment, views all A^C actions as part of the center, so we account for their costs there. To that end, *the cost of a leaf path π^L now is the summed-up cost of its $(A^L|_{F^L} \setminus A^C)$ actions*. By construction, these actions do not affect any factor other than F^L itself.

We will define star-compliant paths, and star-compliant path graphs *CompG*$^\sigma(\pi^C(s), F^L)$, below. For an overview before delving into these details, consider first the *star-decoupled state space* Θ^σ. A *star-decoupled state s* is a center path $\pi^C(s)$ associated for every leaf $F^L \in \mathcal{F}^L$ with the π^C-star-compliant path graph *CompG*$^\sigma(\pi^C(s), F^L)$. The *star-decoupled initial state I^σ* is the empty center path $\pi^C(I^\sigma) = \langle\rangle$. The *star-decoupled goal states* are those s where $\pi^C(s)$ ends in a center goal state, and for every $F^L \in \mathcal{F}^L$ there exists a goal leaf state $s^L \in S^L|_{F^L}$ s.t. $s^L_{|\pi^C(s)|}$ is reachable from $I[F^L]_0$ in *CompG*$^\sigma(\pi^C(s), F^L)$.

The definition of successor states s' of a star-decoupled state s changes more substantially. For forks, these simply were all center paths extending $\pi^C(s)$ by one more center action a^C. This worked due to the absence of leaf preconditions: by definition of "center path", $\mathsf{pre}(a^C)$ was satisfied at the end of $\pi^C(s)$. Given a star factoring instead, the successor states s' still result from extending $\pi^C(s)$ by one more center action a^C, but restricted to those a^C whose leaf preconditions can be satisfied at the end of $\pi^C(s)$. Namely, we require that, for every $F^L \in \mathcal{F}^L$, there exists $s^L \in S^L|_{F^L}$ such that $\mathsf{pre}(a^C)[F^L] \subseteq s^L$ and $s^L_{|\pi^C(s)|}$ is reachable from $I[F^L]_0$ in *CompG*$^\sigma(\pi^C(s), F^L)$.

In our original example, the star-decoupled initial state has two successors, from move(A, l_1, l_2) and move(B, l_3, l_2). In the no-empty example, only move(A, l_1, l_2) is present: Its precondition $p = A$ is reachable from $I[F^L]_0$ given the empty center path. But that is not so for the precondition $p = B$ of move(B, l_3, l_2).

Like for forks, we first identify a notion of compliant paths which captures how plans for the input task Π can be understood as center paths augmented with compliant leaf paths; and we then capture compliant paths in terms of compliant path graphs. However, the notion of "compliance" is now quite a bit more complicated. Given center path π^C and leaf path π^L, we require that (1) the sub-sequences of shared actions in π_L and π_C coincide, and (2) in between, we can schedule π^L at monotonically increasing points alongside π^C s.t. (2a) the center precondition of each leaf action holds in the respective center state and (2b) the F^L precondition of each center action holds in the respective leaf state.

Definition 5 (Star-Compliant Path) *Let Π be an FDR task, $\mathcal{F}$ a star factoring with center F^C and leaves $\mathcal{F}^L$, and π^C a center path. Let π^L be a leaf path for $F^L \in \mathcal{F}^L$. We say that π^L* star-complies *with π^C, also π^L is π^C-*star-compliant*, if the following two conditions hold:*

(1) The sub-sequence of A^C actions in π^L coincides with the sub-sequence of $A^L|_{F^L}$ actions in π^C.

(2) Assume, for ease of notation, dummy $A^C \cap A^L|_{F^L}$ actions added to start and end in each of π^C and π^L. For every pair $\langle a, a' \rangle$ of subsequent $A^C \cap A^L|_{F^L}$ actions in π^L and π^C, there exists an embedding *at $\langle a, a' \rangle$.*

Here, denote the sub-sequence of π^C between a and a' (not including a and a' themselves) by $\langle a^C_1, \dots, a^C_n \rangle$, and the F^C states it traverses by $\langle s^C_0, \dots, s^C_n \rangle$. Denote the sub-sequence of π^L between a and a' by $\langle a^L_1, \dots, a^L_m \rangle$, and the F^L states it traverses by $\langle s^L_0, \dots, s^L_m \rangle$. An embedding *at $\langle a, a' \rangle$ then is a monotonically increasing function $t : \{1, \dots, m\} \mapsto \{0, \dots, n\}$ so that both:*

(a) For every $i \in \{1, \dots, m\}$, $\mathsf{pre}(a^L_i)[F^C] \subseteq s^C_{t(i)}$.

(b) For every $t \in \{1, \dots, n\}$, $\mathsf{pre}(a^C_t)[F^L] \subseteq s^L_{i(t)}$ where $i(t) := \max\{i \mid t(i) < t\}$ (with $\max \emptyset := 0$).

To illustrate this, consider our no-empty example, the center path $\pi^C = \langle$move$(A, l_1, l_2)\rangle$, and the leaf path $\pi^L = \langle$load(A, l_1), unload$(A, l_2)\rangle$. Definition 5 (1) is trivially fulfilled because the sub-sequences it refers to are both empty. For Definition 5 (2), we assume dummy shared actions, $\pi^C = \langle a,$ move$(A, l_1, l_2), a' \rangle$ and $\pi^L = \langle a,$ load(A, l_1), unload$(A, l_2), a' \rangle$. The only pair of subsequent shared actions then is $\langle a, a' \rangle$. We need to find an embedding $t : \{1, 2\} \mapsto \{0, 1\}$ of $\langle a^L_1 =$ load$(A, l_1), a^L_2 =$ unload$(A, l_2)\rangle$ traversing F^L states $\langle s^L_0 = \{p = l_1\}, s^L_1 = \{p = A\}, s^L_2 = \{p = l_2\}\rangle$, into $\langle a^C_1 =$ move$(A, l_1, l_2)\rangle$ traversing F^C states $\langle s^C_0 = \{t_A = l_1\}, s^C_1 = \{t_A = l_2\}\rangle$. Given their center preconditions, we must schedule load(A, l_1) before move(A, l_1, l_2) and unload(A, l_2) behind move(A, l_1, l_2). So the only possibility is $t(1) := 0, t(2) := 1$. Indeed, that is an embedding: For Definition 5 (2a), $\mathsf{pre}(a^L_1)[F^C] = \{t_A = l_1\} \subseteq s^C_{t(1)} = s^C_0$, and $\mathsf{pre}(a^L_2)[F^C] = \{t_A = l_2\} \subseteq s^C_{t(2)} = s^C_1$. For Definition 5

(2b), $i(1) = \max\{i \mid t(i) < 1\} = 1$ because $t(1) = 0$ i. e. $a^L_1 = \text{load}(A, l_1)$ is scheduled before $a^C_1 = \text{move}(A, l_1, l_2)$. So $\mathsf{pre}(a^C_1)[F^L] = \{p = A\} \subseteq s^L_{i(1)} = s^L_1$ as required.

Despite the much more complex definition, the correspondence of compliant paths to plans for the original input planning task Π is as easily seen as for fork factorings. Say π is a plan for Π. The sub-sequence π^C of center actions in π is a center path. For a leaf $F^L \in \mathcal{F}^L$, the sub-sequence π^L of $A^L|_{F^L}$ actions in π is a leaf path. The sub-sequence of $A^C \cap A^L|_{F^L}$ actions in π^L coincides by construction with the sub-sequence of $A^C \cap A^L|_{F^L}$ actions in π^C, so we fulfill Definition 5 (1). Furthermore, between any pair of subsequent shared actions, all F^C preconditions of π^L, and all F^L preconditions of π^C, must be satisfied because π is a plan, so we can read off an embedding fulfilling Definition 5 (2), and π^L is π^C-star-compliant. Vice versa, say center path π^C ends in a goal center state, and can be augmented for every $F^L \in \mathcal{F}^L$ with a π^C-star-compliant leaf path π^L ending in a goal leaf state. Note that, if an action a affects more than one leaf, by the definition of star factorings a must also affect the center, so by Definition 5 (1) the sub-sequences of such actions are synchronized via π^C: They must be identical for every leaf involved, and correspond to the same action occurences in π^C. Hence, sequencing all actions in π^C and every π^L according to the embeddings, we get an executable action sequence π achieving the overall goal in Π. Recall, finally, that we defined the cost of leaf paths to account only for those actions affecting just the leaf in question and nothing else. So, in both directions above, the cost of π equals the summed-up cost of the center path and leaf paths. We get that *the plans for Π are in one-to-one correspondence with center paths augmented with compliant leaf paths.*

We finally show how to capture π^C-star-compliant paths in terms of the weighted graphs *CompG*$^\sigma(\pi^C(s), F^L)$ we maintain alongside search over center paths in Θ^σ:

Definition 6 (Star-Compliant Path Graph) *Let Π be an FDR task, $\mathcal{F}$ a star factoring with center F^C and leaves $\mathcal{F}^L$, and $\pi^C = \langle a^C_1, \dots, a^C_n \rangle$ a center path traversing center states $\langle s^C_0, \dots, s^C_n \rangle$. The* π^C-star-compliant path graph *for a leaf $F^L \in \mathcal{F}^L$, denoted CompG$^\sigma(\pi^C, F^L)$, is the arc-labeled weigthed directed graph whose vertices are $\{s^L_t \mid s^L \in S^L|_{F^L}, 0 \leq t \leq n\}$, and whose arcs are as follows:*

(i) $s^L_t \xrightarrow{a^L} {s'}^L_t$ with weight $c(a^L)$ whenever $s^L, {s'}^L \in S^L|_{F^L}$ and $a^L \in A^L|_{F^L} \setminus A^C$ s.t. $\mathsf{pre}(a^L)[F^C] \subseteq s^C_t$, $\mathsf{pre}(a^L)[F^L] \subseteq s^L$, and $s^L[\![a^L]\!] = {s'}^L$.

(ii) $s^L_t \xrightarrow{0} {s'}^L_{t+1}$ with weight 0 whenever $s^L, {s'}^L \in S^L|_{F^L}$ s.t. $\mathsf{pre}(a^C_t)[F^L] \subseteq s^L$ and $s^L[\![a^C_t]\!] = {s'}^L$.

Item (i) is a benign change of Definition 3. Exactly as before, within each time step t the arcs correspond to those leaf-only actions whose center precondition is enabled at t. The only difference is that we need to explicitly exclude actions a^L affecting also the center (which for fork factorings cannot happen anyway). Item (ii) differs more substantially. Intuitively, whereas for fork factorings the $t \to t + 1$ arcs simply stated that whichever leaf state we achieved before will survive the center action a^C_t (which could neither rely on, nor affect, the leaf), these arcs now state that the surviving leaf states are only those which comply with a^C_t's precondition, and will be mapped to possibly different leaf states by a^C_t's effect. Note that, if a^C_t has no precondition on F^L, then all leaf states survive, and if a^C_t has no effect on F^L, then all leaf states remain the same at $t+1$. If both is the case, then we are back to exactly the arcs (ii) in Definition 3.

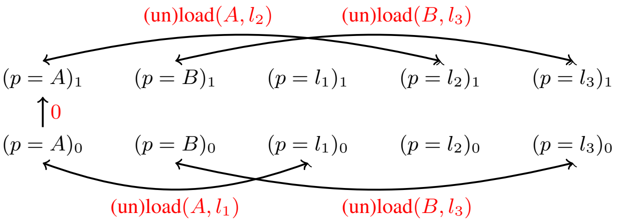

Figure 3: The star-compliant path graph for $\pi^C = \langle \text{move}(A, l_1, l_2) \rangle$ in our no-empty example.

For our no-empty task and $\pi^C = \langle \text{move}(A, l_1, l_2) \rangle$, the π^C-star-compliant path graph is as shown in Figure 3.

Note the (only) difference to Figure 2: From time 0 to time 1, the only (ii) arc we have now is that from $(p = A)_0$ to $(p = A)_1$. This is because move(A, l_1, l_2) now has precondition $p = A$, so all other values of p do not comply with the center action being applied at this time step.

Lemma 2 *Let Π be an FDR task, $\mathcal{F}$ a star factoring with center F^C and leaves $\mathcal{F}^L$, and π^C a center path. Let $F^L \in \mathcal{F}^L$, and $s^L \in S^L|_{F^L}$. Then the cost of a cheapest π^C-star-compliant leaf path π^L ending in s^L equals the cost of a cheapest path from $I[F^L]_0$ to s^L_n in CompG$^\sigma(\pi^C, F^L)$.*

Proof: Consider first a π^C-star-compliant leaf path $\pi^L = \langle a^L_1, \dots, a^L_m \rangle$ for leaf $F^L \in \mathcal{F}^L$. By Definition 5 (1), the $A^C \cap A^L|_{F^L}$ sub-sequences in π^C and π^L coincide. Scheduling these at the respective time steps $t \to t + 1$ in *CompG*$^\sigma(\pi^C, F^L)$, corresponding (ii) arcs must be present by construction. In between each pair of such actions, by Definition 5 we have embeddings t mapping the respective sub-sequence of π^L to that of π^C. Schedule each π^L action at its time step assigned by t. Then corresponding (i) arcs must be present by Definition 5 (2a). By Definition 5 (2b), if a π^C action here relies on an F^L precondition, then the corresponding leaf state satisfies that precondition so we have the necessary (ii) arc. Overall, we obtain a path π from $I[F^L]_0$ to s^L_n in *CompG*$^\sigma(\pi^C, F^L)$, and clearly the cost of π accounts exactly for the F^L-only actions on π^L, as needed.

Vice versa, consider any path π from $I[F^L]_0$ to s^L_n in *CompG*$^\sigma(\pi^C, F^L)$. Removing the time indices of the vertices on π, and removing those (ii) arcs $s^L_t \xrightarrow{0} {s'}^L_{t+1}$ where $s^L_t = {s'}^L_{t+1}$, clearly we obtain a π^C-star-compliant leaf path π^L ending in s^L, whose cost equals that of π.

So the π^C-star-compliant leaf paths ending in s^L are in one-to-one correspondence with the paths from $I[F^L]_0$ to s^L_n in *CompG*$^\sigma(\pi^C, F^L)$, showing the claim. □

Overall, goal paths in the star-decoupled state space Θ^σ correspond to center goal paths augmented with star-compliant leaf goal paths, which correspond to plans for the original planning task Π, of the same cost. So (optimal) search in Θ^σ is a form of (optimal) planning for Π.

Heuristic Search

GH show how standard classical planning heuristics, and standard search algorithms, can be applied to fork-decoupled search. All these concepts remain intact for star topologies; one issue requires non-trivial attention. (For space reasons, we omit details and give a summary only.)

A *heuristic* for Θ^σ is a function from star-decoupled states into $\mathbb{R}^{0+} \cup \{\infty\}$. The *star-perfect* heuristic, $h^{\sigma *}$, assigns to any s the minimum cost for completing s, i. e., reaching a star-decoupled goal state plus embedding compliant goal leaf paths. A heuristic h is *star-admissible* if $h \leq h^{\sigma *}$.

Given an FDR task Π and a star-decoupled state s, one can construct an FDR task $\Pi^\sigma(s)$ so that computing any admissible heuristic h on $\Pi^\sigma(s)$ delivers a star-admissible heuristic value for s. $\Pi^\sigma(s)$ is like Π except for the initial state (center state of s, initial state for the leaves), and that new actions are added allowing to achieve each leaf state at its price in s.

Star-decoupled goal states are, as GH put it, *goal states with price tags*: Their path cost accounts only for the center moves, and we still have to pay the price for the goal leaf paths. In particular, $h^{\sigma *}$ is *not* 0 on star-decoupled goal states. We can obtain a standard structure Θ' for search as follows. Introduce a new goal state G. Give every star-decoupled goal state s an outgoing transition to G whose cost equals the summed-up cost of cheapest compliant goal leaf paths in s. Given a heuristic h for Θ^σ, set $h(G) := 0$.

The generalization to star-decoupling incurs one important issue, not present in the special case of fork factorings. If center moves require preconditions on leaves, then we should "buy" these preconditions immediately, putting their price into the path cost g, because otherwise we lose information during the search. For illustration, in our no-empty example, say the goal is $t_A = l_2$ instead of $p = l_3$, and consider the star-decoupled state s after applying move(A, l_1, l_2). Then $t_A = l_2$ is true, $g = 1$, and h^* on the compiled FDR task $\Pi^\sigma(s)$ returns 0 because the goal is already true. But $h^{\sigma *}(s) = 1$ and the actual cost of the plan is 2: We still need to pay the price for the precondition $p = A$ of move(A, l_1, l_2). This is not captured in $\Pi^\sigma(s)$ because it is needed *prior* to s only. The solution is to perceive this "price" as a "cost" already committed to. In our modified structure Θ', when applying a center action a^C to star-decoupled state s, we set the local cost of a^C (its cost specifically at this particular position in Θ') to $\mathsf{cost}(a_t^C) + \sum_{F^L} g(F^L)$. Here, $g(F^L)$ is the minimum over the price in s of those $s^L \in S^L|_{F^L}$ that satisfy a^C's precondition. Intuitively, to apply a^C, we must first buy its leaf preconditions. To reflect that $g(F^L)$ has already been paid, the respective (ii) arcs in *CompG*$^\sigma(\pi^C(s), F^L)$ are assigned weight $-g(F^L)$. In our example above, the path cost in s is $g = 2$ giving us the correct $g + h = 2$. The "0" arc in Figure 3 is assigned weight -1, so that the overall cost of the compliant path for p will be 0 (as the only action we need to use has already been paid for by the center move).

Any (optimal) standard heuristic search algorithm X on Θ' yields an optimal heuristic search algorithm for Θ^σ, which we denote *Star-Decoupled X (SDX)*.

Experiments

Our implementation is in FD (Helmert 2006), extending that for fork decoupling by GH. We ran all international planning competition (IPC) STRIPS benchmarks ('98–'14), on a cluster of Intel E5-2660 machines running at 2.20 GHz, with time (memory) cut-offs of 30 minutes (4 GB).

Our experiments are preliminary in that we perform only a very limited exploration of factoring strategies. Factoring strategy design for star topologies is, in contrast to fork topologies, quite challenging. The space of star factorings includes arbitrary two-subset partitions of single-SCC causal graphs, where fork factorings do not exist at all. Even for the simplest possible optimization criterion, maximizing the number of leaves in a strict star factoring, finding an optimal factoring is **NP**-complete (this follows by a straightforward reduction from Maximum Independent Set (Garey and Johnson 1979)). An additional complication is that leaves may be "frozen": As we need to branch over all actions affecting the center, for a leaf F^L to yield a state space size reduction there must be at least one action affecting *only* F^L (not affecting the center). For example, in IPC Visit-All, while the robot position may naturally be viewed as the "center" and each "visited" variable as a leaf, every leaf-moving action also affects the center so nothing is gained.

We shun this complexity here, leaving its comprehensive exploration to future work, and instead design only two simple strict-star factoring strategies by direct extension of GH's fork factoring strategy. That strategy works as follows.

Denote by $\mathcal{F}^{\text{SCC}}$ the factoring whose factors are the SCCs of *CG*. View the interaction graph *IG*$(\mathcal{F}^{\text{SCC}})$ over these SCCs as a DAG where the root SCCs are at the top and the leaf SCCs at the bottom. Consider the "horizontal lines" $\{T, B\}$ (top, bottom) through that DAG, i. e., the partitions of V where every $F \in \mathcal{F}^{\text{SCC}}$ is fully contained in either of T or B, and where the only arc in *IG*$(\{T, B\})$ is $(T \rightarrow B)$. Let $\mathcal{W}$ be the set of weakly connected components of $\mathcal{F}^{\text{SCC}}$ within B. Then a fork factoring $\mathcal{F}$ is obtained by setting $F^C := T$ and $\mathcal{F}^L := \mathcal{W}$. Any fork factoring can be obtained in this manner, except the *redundant* ones where some $F^L \in \mathcal{F}^L$ contains several weakly connected components.

GH's strategy moves the horizontal line upwards, from leaves to roots in *IG*$(\mathcal{F}^{\text{SCC}})$, in a greedy fashion, thereby generating a sequence $\mathcal{F}_1, \dots, \mathcal{F}_k$ of fork factorings. They select the factoring $\mathcal{F}_i$ whose number of leaf factors is maximal, and whose index i is minimal among these factorings. The rationale behind this is to maximize the number of leaves (the amount of conditional independence) while keeping these as small as possible (reducing the runtime overhead). If $k = 0$ (no horizontal line exists i. e. *CG* is a single SCC), or $\mathcal{F}_i$ has a single leaf only, then GH *abstain* from solving the input task. The rationale is that, in GH's experiments, single-leaf factorings hardly ever payed off.

	(A): Coverage												(B): Evaluations: Improvement factor relative to GBFS											(C): Runtime: Improvement factor relative to GBFS										
	X-shape						inverted fork			fork			X-shape																					
													no preferred operators				with preferred operators							no preferred operators				with preferred operators						
		npo		po				npo	po		npo	po		SDGBFS				SDGBFS			LAMA				SDGBFS				SDGBFS			LAMA		
	#	base	sd	base	sd	LAMA	#	sd	sd	#	sd	sd	#	$\sum$D	GM	max	#	$\sum$D	GM	max	$\sum$D	GM	max	#	$\sum$D	GM	max	#	$\sum$D	GM	max	$\sum$D	GM	max
Childsna	20	0	0	3	+1	-3	20	0	+1	0			0				0							0				0						
Depots	22	14	-1	18	-1	+3	22	-1	-1	0			12	0.5	1.1	14.0	17	1.5	2.1	54.5	3.4	2.3	299	10	0.3	0.3	5.7	14	1.0	0.7	22.5	0.8	1.3	91.0
Driver	20	18	+1	20	-2	0	0			20	+1	-2	17	1.2	2.2	8.3	18	0.1	0.9	4.0	0.7	0.9	21.0	5	0.3	0.4	3.3	12	0.0	0.2	1.0	0.5	0.6	2.1
Elev08	30	30	0	30	0	0	30	0	0	0			30	5.5	5.4	18.1	30	4.2	3.5	12.1	1.0	1.1	3.8	24	2.0	1.5	4.0	20	1.4	1.1	2.0	0.6	0.6	1.4
Elev11	20	18	+2	20	0	0	20	+2	0	0			18	12.6	7.5	46.8	20	11.5	5.6	37.3	1.8	1.1	4.3	18	4.8	2.7	18.5	20	4.7	1.8	13.0	1.3	0.7	2.8
Floor11	20	6	-5	6	-4	0	20	-5	-4	0			1	0.4	0.4	0.4	2	0.7	0.7	0.7	0.8	0.8	1.0	1	0.1	0.1	0.1	2	0.1	0.1	0.1	0.2	0.2	0.3
Floor14	20	2	-2	2	0	0	20	-2	0	0			0				2	0.6	0.6	0.7	0.9	0.9	1.1	0				2	0.1	0.1	0.1	0.2	0.2	0.3
Log00	28	28	0	28	0	0	28	0	0	28	0	0	28	11.8	9.4	18.8	28	5.2	4.7	6.8	0.8	0.9	1.0	0				0						
Log98	35	26	+9	35	0	0	35	+9	0	35	+9	0	26	37.3	12.7	60.0	35	49.4	3.5	177	9.0	1.1	28.1	16	19.0	6.7	34.9	24	19.1	1.5	65.8	4.3	0.8	18.0
Mico	145	145	0	145	0	0	0			145	0	0	145	2.4	2.2	4.4	145	4.3	3.6	8.4	1.0	1.0	1.8	26	1.1	1.1	1.6	45	1.0	1.0	1.2	0.5	0.6	0.9
Mystery	2	0	0	1	0	0	2	0	0	0			0				1	0.9	0.9	0.9	1.4	1.3	1.3	0				1	0.6	0.6	0.6	1.0	1.0	1.0
NoMy	20	9	+10	10	+9	+3	0			20	+10	+9	9	141	34.0	1250	10	43K	33.5	15M	1697	4.5	135K	5	2.6	2.2	5.9	8	1033	3.8	12K	468	2.1	10K
Pathw	30	11	+2	20	0	+4	0			29	+2	0	11	12.7	3.8	26.5	20	1.4	1.5	1.8	0.6	0.7	1.2	7	3.5	1.3	10.7	10	0.4	0.6	1.0	0.3	0.4	1.0
PSR	3	3	0	3	0	0	0			3	0	0	3	1.6	1.7	1.9	3	1.3	1.2	1.6	1.0	1.0	1.0	0				0						
Rovers	40	23	-1	40	0	0	38	+5	-1	40	-1	0	22	1.2	1.7	12.3	40	1.3	1.7	3.1	0.7	0.8	1.8	12	0.4	0.5	1.7	22	0.6	0.7	1.0	0.3	0.4	0.8
Satell	36	30	+3	36	0	0	34	+1	0	36	+3	0	30	2.7	2.0	38.7	36	1.8	1.3	6.8	0.4	0.5	1.5	19	1.1	1.8	21.9	25	1.9	1.0	3.9	0.1	0.3	0.8
TPP	29	22	+1	29	0	0	26	-5	0	27	+1	0	22	3183	454	31K	29	0.1	3.4	41.9	0.8	0.8	1.8	14	95.8	52.7	987	17	0.0	0.2	1.3	0.5	0.6	0.9
Transp08	30	16	+14	28	+2	+2	30	+14	+2	0			16	693	35.7	2226	28	305	21.0	4360	1.0	1.0	3.3	10	135	17.6	262	22	18.0	2.0	156	0.6	0.7	1.7
Transp11	20	0	+20	11	+9	+7	20	+20	+9	0			0				11	500	114	4360	1.0	1.1	8.9	0				11	19.0	5.2	141	0.7	0.7	4.7
Transp14	20	0	+20	6	+14	+9	20	+20	+14	0			0				6	168	130	407	1.1	1.1	6.9	0				6	8.7	7.4	23.9	0.6	0.8	4.4
Wood08	26	26	0	26	0	0	25	0	0	25	0	0	26	0.4	1.2	121	26	1.2	2.3	111	7.8	14.8	145	20	0.1	0.2	2.1	22	0.2	0.3	7.9	2.1	2.4	6.3
Wood11	19	18	0	19	0	0	18	+1	0	18	+1	0	2	0.4	0.5	4.3	19	1.2	1.2	3.0	11.8	15.8	53.8	16	0.0	0.1	0.5	18	0.0	0.1	0.2	2.1	2.1	4.9
Zeno	20	20	0	20	0	0	18	0	0	20	0	0	20	29.3	12.1	103	20	4.6	3.0	6.3	0.8	0.8	1.3	7	4.2	3.8	10.6	7	0.9	0.8	1.1	0.4	0.4	0.7
$\sum$	655	465	+73	556	+28	+25	426	+59	+20	446	+26	+7																						

Table 1: Results in satisficing planning with h^{FF}. Each table section fixes a factoring strategy (2nd row from top), and uses *only the instances which that strategy does not abstain on*; of these, (B) uses the subset of commonly solved ones (with vs. without preferred operators), (C) the same but excluding ones commonly solved in ≤ 0.1 seconds. The respective numbers of underlying instances are shown in columns "#". GBFS: Greedy Best-First Search; npo: no preferred ops; po: with preferred ops (FD's dual-queue search); base: baseline (GBFS on standard state space); sd: star-decoupled base; $\sum$D: factor over the per-domain sums; GM/max: Geometric mean/maximum over the per-instance factors; K: thousand; M: million.

We design two new (non-fork) strategies, *inverted forks* and *X-shape*. The former is exactly GH's strategy but inverting the direction of the arcs in the causal graph. The latter runs GH's fork factoring first, and thereafter runs inverted forks on the fork center component F^C. If an inverted-fork leaf F^L has an outgoing arc into a fork leaf, then F^L is included into F^C. We abstain if no factoring exists or if the selected factoring has a single leaf only. Note that this still abstains on single-SCC causal graphs, and that frozen leaves cannot occur as neither forks nor inverted forks allow leaf-affecting actions to affect the center. The strategies take negligible runtime (rounded to 0.00 in most cases, much faster than FD's pre-processes in the few other cases).

For optimal planning (with LM-cut (Helmert and Domshlak 2009)), while GH reported dramatic gains using fork factoring, our new factoring strategies do not improve much upon these gains. Inverted forks do sometimes help, most notably in Satellite where Star-Decoupled A* (SDA*) with inverted fork factoring solves 3 more instances than each of A* and fork-factoring SDA*, reducing evaluations on commonly solved instances by up to two orders of magnitude. But such cases are rare. It remains future work to explore more sophisticated star-factoring strategies for optimal planning. Here, we focus on satisficing planning where even our current simple factoring strategies yield good results.

Consider Table 1. X-shape factoring abstains much less than each of the "base strategies", forks and inverted forks, on its own. X-shape factoring substantially improves coverage, overall and in several domains, without preferred operators, and does so in NoMystery and Transport with preferred operators. It even beats LAMA (Richter and Westphal 2010) overall, thanks to excelling in Transport. Indeed, with preferred operators, Transport is solved almost instantly, with a maximum (average) of 69 (36.1) state evaluations and 14.7 (2.7) seconds runtime. Without preferred operators, the average is 1236.8 evaluations and 127.1 seconds. Substantial reductions of evaluations, with corresponding smaller but still significant reductions of runtime, are obtained also in Elevators, Logistics, NoMystery, Pathways, TPP, and Zenotravel, plus smaller improvements in various other cases.

The strength of X-shape factoring is inherited from fork factoring in NoMystery and Pathways, where inverted fork factoring abstains. It stems from inverted fork factoring, and is thus entirely thanks to the new techniques introduced herein, in Elevators and Transport where fork factoring abstains. For Logistics, TPP, and Zenotravel, where neither base strategy abstains, our current configuration of X-shape factoring uses exactly the fork factorings, because the fork strategy is run first and thus given a "preference". To illustrate, in Logistics, the fork makes each package a leaf. The inverted fork would make the trucks leaves, but they have outgoing arcs to the packages so are re-included into the center. Symmetrically, if the X-shape ran the inverted fork strategy first, it would end up with exactly the inverted fork factorings. In Logistics98, these have a consistent advantage ($\sum$D evaluations improvement factor over forks is 6.4 with preferred operators). In Logistics00 they have a consistent small disadvantage ($\sum$D factor 0.9). In TPP and Zenotravel, there is significant per-instance variance, up to 14 times worse respectively 537 times better in TPP, and up to 2 times worse respectively 5 times better in Zenotravel. So there may be room for improvement by combining forks and inverted forks in a less simplistic manner.

As we are mainly interested in speed, the above considers uniform action costs. The results for non-uniform action costs are very similar, except in Elevators where both the baseline and X-shape factoring get worse, in Transport where only the baseline gets worse, and in Woodworking where only X-shape factoring without preferred operators gets worse. The overall coverage gain is then 49 without preferred operators, and 35 with preferred operators.

Conclusion

For our current simple factoring strategies, the upshot from our experiments is that the main empirical advantage of star topologies over forks (as far as IPC benchmarks are concerned) stems from the Transport domain, which this technique "kills" completely. The most important research question in planning remains the exploration of factoring strategies. We need methods able to identify sophisticated star topologies, and we need to understand the strengths of different factorings as a function of problem structure.

Beyond planning, an interesting opportunity is to apply star-topology decoupling to star-topology systems. Verification & synthesis appear especially relevant, possibly with adversarial or probabilistic extensions to star-decoupling. Some classical puzzles may also be amenable; e. g., in the Peg Solitaire game, one can take the "center" to be the middle of the board and the "leaves" to be its peripheral parts.

Acknowledgments. We thank the anonymous reviewers, whose comments helped to improve the paper. Daniel Gnad's travel to SoCS'15 was partially supported by the AI section of the German Informatics Society (GI).

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. 2003. Structure and complexity in planning with unary operators. *Journal of Artificial Intelligence Research* 18:315–349.

Brafman, R. I., 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. 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.

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.

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.

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

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

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.

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

Hoffmann, J. 2011. Analyzing search topology without running any search: On the connection between causal graphs and h^+. *Journal of Artificial Intelligence Research* 41:155–229.

Jonsson, P., and Bäckström, C. 1995. Incremental planning. In *European Workshop on Planning*.

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

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.