Decoupled State Space Search – Dissertation Abstract

Daniel Gnad
Saarland University
Saarbrücken, Germany
gnad@cs.uni-saarland.de

Abstract

Decoupled State Space Search is a recent approach to exploiting problem structure in classical planning. The particular structure needed is a star topology, with a single *center* component interacting with multiple *leaf* components. All interaction of the leaves with the rest of the problem has to be via the center. Given this kind of problem decomposition, we have showed that search on this *reformulated state space* can be exponentially more efficient than standard search. However, there do also exist cases in which decoupled search has to spend exponentially *more* effort to solve a task. We want to tackle this issue by combining decoupled search with different known search enhancement techniques, such as partial-order reduction, symmetry reduction, or dominance pruning. Presumably, these can be nicely combined with our new approach, such that we can prevent the exponential blow-up. Decoupled search is not restricted to classical planning, though. Its principles apply to all kinds of (heuristic) search problems, like, e. g., in Model Checking.

Introduction

In classical planning, heuristic search is a popular approach to solve a variety of input problems. The solution of such a planning task takes the form of a path, i. e., a sequence of transitions that lead from a given start state to a state satisfying certain goal conditions. To find such a path, (possibly huge) deterministic transition system, the task's *state space*, need to be explored. Inherent in this way of finding solutions to planning problems is the issue of state space explosion. We propose decoupled state space search (Gnad and Hoffmann 2015; Gnad *et al.* 2015) to solve this problem. Decoupled search can be seen as a form of *factored planning* (e. g., (Amir and Engelhardt 2003; Brafman and Domshlak 2006; 2008; 2013; Fabre *et al.* 2010)) , that restricts the interaction between the factors to take the form of a *star*, with a single center factor that interacts with multiple leaf factors. All interaction of a leaf with another factor has to be via the center. Hereby, decoupled search exploits a kind of conditional independence between the leaves – given a fixed center path π^C, *compliant* leaf paths can be scheduled independently along π^C. This allows for an efficient search and solution reconstruction, since complex cross-factor dependencies do not have to be resolved, which can make other factored planning approaches infeasible in practice. In our experiments, we observed that the decoupled state space, oftentimes is exponentially smaller than the standard state space.

There is also bad news, however. Although in most of our experiments we see an exponential reduction of the state space size under decoupled search, the state space can also get exponentially larger. This is so because of the special structure of the leaves, that "remember" the center path leading to the current decoupled state. Thus, when reaching the same state via different paths, decoupled search treats all these states as if they were different, leading to the blow-up. One possible means to circumvent this issue are dominance pruning methods. The simple method employed by (Gnad and Hoffmann 2015) already suffices to guarantee that the decoupled state space – which in principle can grow to infinite size – stays finite. Future work is going to derive more elaborate pruning methods that are more effective in reducing the size of the decoupled state space; the eventual goal being to upper bound its size by that of the standard state space.

Besides, many search enhancement techniques that have been proposed for standard state space search can probably also be deployed in the decoupled setting. Prominent topics in standard search are for example partial-order reduction, or symmetry, and dominance pruning.

Further more, although our theoretical framework allows for general star-shape factorings, in practice we only use fork and inverted-fork like structures. One of the reasons why we stuck to this is the complexity of computing general star factorings. Even if the only objective is the maximization of the number of leaf factors, obtaining such a factoring is **NP**-hard.

The rest of this work is organized as follows. The next chapter gives a brief summary of the relevant definitions of decoupled search as provided by (Gnad and Hoffmann 2015; Gnad *et al.* 2015). The reader familiar with decoupled search is invited to skip this chapter and proceed to *Future Work*, which introduces several lines of future research. We conclude with a brief summary.

Decoupled Search

Background

Our prior work has introduced Decoupled Search using 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 $s[\![\langle a_1, \ldots, a_n \rangle]\!]$. A *plan* for Π is an action sequence s.t. $G \subseteq I[\![\langle a_1, \ldots, a_n \rangle]\!]$. The plan is *optimal* if its summed-up cost is minimal among all plans for Π.

To define factorings, we need the notion of the *causal graph* (e. g. (Knoblock 1994; Jonsson and Bäckström 1995; Brafman and Domshlak 2003; Helmert 2006)) using 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 $(v \to v')$, 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 $(v \to v')$ 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 $(v \to v')$ 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.

Given the required notation, we can now define a factoring as a partition of the variables V into non-empty subsets $\mathcal{F}$, called factors.

Definition 1 (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 $\{(F^C \to F^L), (F^L \to F^C) \mid F^L \in \mathcal{F} \setminus \{F^C\}\}$.*

Note that *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 1 does not apply any restrictions if there is a single leaf only. That said, it is not clear whether single-leaf factorings are useful in practice.

Example 1 *As an illustrative example, consider a transportation task with one package p, and two trucks t_A, t_B moving along three locations l_1, l_2, l_3 arranged in a line. The FDR planning task $\Pi = \langle V, A, I, G \rangle$ is 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) with precondition $\{t_x = y, p = x\}$ 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) with 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) with precondition $\{t_x = y, p = x\}$ and effect $\{p = y\}$, where $x \in \{A, B\}$ and $y \in \{l_1, l_2, l_3\}$.*

Observe that a truck can only move if the package is currently inside it. The causal graph is shown in Figure 1.

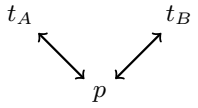

Figure 1: The causal graph of the example.

In this task, several factorings are possible. Consider, e. g., $\mathcal{F}_1 = \{\{t_A, t_B\}, \{p\}\}$, $\mathcal{F}_2 = \{\{t_A\}, \{t_B\}, \{p\}\}$, or $\mathcal{F}_3 = \{\{t_A\}, \{t_B, p\}\}$, all of which are clearly star factorings (though for F_2 only if we set $F^C = \{p\}$).

We are now defining the state space of a decoupled planning task. In contrast to standard search, decoupled search only branches over the center actions, enumerating what each leaf factor can do, separately. The *center actions* A^C are all those actions affecting the center. The *leaf actions* $A^L|_{F^L}$ for $F^L \in \mathcal{F}^L$ are all those actions affecting F^L. Observe that A^C *and* $A^L|_{F^L}$ *are not disjoint*, as the same action

may affect both A^C and $A^L|_{F^L}$. A leaf path is a sequence of leaf actions applicable to I when ignoring all center preconditions. A center path is a sequence of center actions applicable to I when ignoring all leaf preconditions.

After applying a center action to a state s, we update what is called the *pricing function* of s, $\mathsf{prices}[s]$. $\mathsf{prices}[s]$ assigns each leaf state of a leaf factor a price, representing the summed-up leaf action cost one would have to spend to reach the state from the initial state of the leaf factor. The update is performed for every reachable leaf state in s. We apply all leaf actions applicable given the center preconditions of $s[\![a]\!]$, updating $\mathsf{prices}[s[\![a]\!]]$ accordingly. More formally, the decoupled state space is defined as follows:

Definition 2 (Decoupled State Space) *Let Π be an FDR task, and $\mathcal{F}$ a star factoring with center F^C and leaves $\mathcal{F}^L$. A* decoupled state *s is a triple $\langle \mathsf{center}[s], \pi^C[s], \mathsf{prices}[s] \rangle$ where $\mathsf{center}[s]$ is a center state, $\pi^C[s]$ is a center path ending in $\mathsf{center}[s]$, and $\mathsf{prices}[s]$ is a* pricing function*, $\mathsf{prices}[s] : S^L \mapsto \mathbb{R}^{0+} \cup \{\infty\}$, mapping each leaf state to a non-negative* price*. The* decoupled state space *is a labeled transition system $\Theta^{\mathcal{F}}_{\Pi} = \langle S^{\mathcal{F}}, A^C, T^{\mathcal{F}}, I^{\mathcal{F}}, S^{\mathcal{F}}_G \rangle$ as follows:*

(i) $S^{\mathcal{F}}$ is the set of all decoupled states.

(ii) A^C, the set of center actions, gives the transition labels.

(iii) $T^{\mathcal{F}}$ is the set of transitions, with $(s \xrightarrow{a^C} t) \in T^{\mathcal{F}}$ if: $a^C \in A^C$; $\pi^C[s] \circ \langle a^C \rangle = \pi^C[t]$; $\mathsf{pre}(a^C)[F^C] \subseteq \mathsf{center}[s]$ and $\mathsf{center}[s][\![a^C]\!] = \mathsf{center}[t]$; for every $F^L \in \mathcal{F}^L$ where $\mathsf{pre}(a^C)[F^L] \neq \emptyset$, there exists $s^L \in S^L|_{F^L}$ s.t. $\mathsf{pre}(a^C)[F^L] \subseteq s^L$ and $\mathsf{prices}[s](s^L) < \infty$; and, for every leaf $F^L \in \mathcal{F}^L$ and leaf state $s^L \in S^L|_{F^L}$, $\mathsf{prices}[t](s^L)$ is the cost of a cheapest path from $I[F^L]_0$ to s^L_n in $\mathsf{CompG}_{\Pi}[\pi^C[t], F^L]$, where $n := |\pi^C[t]|$.

(iv) $I^{\mathcal{F}}$ is the decoupled initial state*, where $\mathsf{center}[I^{\mathcal{F}}] := I[F^C]$, $\pi^C[I^{\mathcal{F}}] := \langle\rangle$, and, for every leaf $F^L \in \mathcal{F}^L$ and leaf state $s^L \in S^L|_{F^L}$, $\mathsf{prices}[I^{\mathcal{F}}](s^L)$ is the cost of a cheapest path from $I[F^L]_0$ to s^L_0 in $\mathsf{CompG}_{\Pi}[\langle\rangle, F^L]$.*

(v) $S^{\mathcal{F}}_G$ are the decoupled goal states *s_G, where $\mathsf{center}[s_G]$ is a center goal state and, for every $F^L \in \mathcal{F}^L$, there exists a leaf goal state $s^L \in S^L|_{F^L}$ s.t. $\mathsf{prices}[s_G](s^L) < \infty$.*

We refer to paths $\pi^{\mathcal{F}}$ in $\Theta^{\mathcal{F}}_{\Pi}$ as decoupled paths*. A* solution *for $s \in S^{\mathcal{F}}$ is a decoupled path (denoted* GlobalPlan*) from s to some $s_G \in S^{\mathcal{F}}_G$. A solution for $\Theta^{\mathcal{F}}_{\Pi}$ is a called a solution for $I^{\mathcal{F}}$. A decoupled state, respectively $\Theta^{\mathcal{F}}_{\Pi}$, is* solvable *if it has a solution.*

Example 2 *In our example, when using the factoring $\mathcal{F} = \{\{p\}, \{t_a, t_B\}\}$ with center factor $F^C = \{t_A, t_B\}$, the initial state of the decoupled state space has only a single successor, resulting from $a^C = move(A, l_1, l_2)$. The precondition $p = A$ of $move(A, l_1, l_2)$ is reachable from $I[F^L]_0$ given the empty center path, but that is not so for the precondition $p = B$ of $move(B, l_3, l_2)$. This reflects the fact that, in the initial state of the task, we can move* only *the package (not moving a truck i. e. the center) so that $move(A, l_1, l_2)$ becomes applicable, but we cannot make $move(B, l_3, l_2)$ applicable in this manner.*

We don't provide the details of *compliant paths* and *compliant path graphs*, here, since this is out of scope for the presented work. Instead, we briefly summarize the intuition behind the notion of compliance.

A decoupled state s is a center path $\pi^C(s)$ associated for every leaf $F^L \in \mathcal{F}^L$ with the *π^C-compliant path graph* $\mathsf{CompG}_{\Pi}[\pi^C(s), F^L]$. Given $\pi^C(s)$ and a leaf path π^L, for π^L to be *compliant* with $\pi^C(s)$ we require that (1) the subsequences 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. The compliant path graph of s for a leaf F^L keeps track of *all* leaf paths in the leaf state space of F^L compliant with $\pi^C(s)$. In practice, we do not store the compliant path graphs for each state, but only the corresponding pricing functions.

For illustration, consider Example 3. The arcs from one layer to the next state that the surviving leaf states are only those which comply with $\mathrm{move}(A, l_1, l_2)$'s precondition, and will be mapped to possibly different leaf states by $\mathrm{move}(A, l_1, l_2)$'s effect. Within each layer the arcs correspond to those leaf-only actions whose center precondition is enabled at t. Note that, if $\mathrm{move}(A, l_1, l_2)$ had no precondition on F^L, then all leaf states would survive, and since $\mathrm{move}(A, l_1, l_2)$ has no effect on F^L, all leaf states remain the same at $t + 1$.

Example 3 *Consider again our illustrative example, using the factoring $\mathcal{F} = \{\{p\}, \{t_a, t_B\}\}$ with center factor $F^C = \{t_A, t_B\}$ and the center path $\pi^C = \langle move(A, l_1, l_2) \rangle$. The π^C-compliant path graph is shown in Figure 2.*

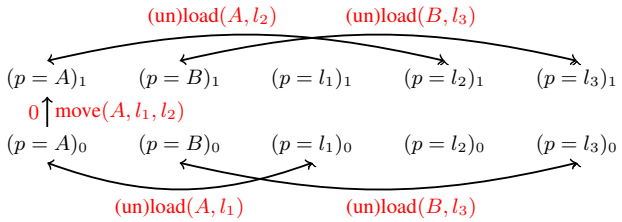

Figure 2: The compliant path graph for $\pi^C = \langle \mathrm{move}(A, l_1, l_2) \rangle$ in our illustrative example.

From layer 0 to layer 1, the only arc we have is that from $(p = A)_0$ to $(p = A)_1$. This is because $move(A, l_1, l_2)$ has precondition $p = A$, so all other values of p do not comply with the center action being applied at this layer, and are excluded from the compliant paths. Note that the arc has a weight of 0, because we do not account for the cost of center actions in the compliant path graph.

In our previous work, we showed that the plans for Π are in one-to-one correspondence with center paths augmented with compliant leaf paths. 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. 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, and π^L is π^C-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-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 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 occurrences in π^C.

Overall, goal paths in the decoupled state space $\Theta^{\mathcal{F}}$ correspond to center goal paths augmented with compliant leaf goal paths, which correspond to plans for the original planning task Π, of the same cost. So (optimal) search in $\Theta^{\mathcal{F}}$ is a form of (optimal) planning for Π.

Heuristic Functions

In decoupled search, two different kinds of heuristic functions are of interest. *Center heuristics* h^C that estimate the remaining cost the center has to spend to reach the goal, and *star heuristics* $h^{\mathcal{S}}$ that estimate the overall remaining cost. We say that h is *center-admissible* if $h \leq h^{C*}$, and *star-admissible* if $h \leq h^{\mathcal{S}*}$. The conceptual distinction between h^C and $h^{\mathcal{S}}$ lies in that h^C cares only about how much work is left for the center factor, i. e., the cost of a center path sufficient to enable every leaf to reach its goal *somehow*. In contrast, $h^{\mathcal{S}}$ accounts for the combined cost of center and leaves, i. e. for the best extension of our current center path into an overall decoupled plan. We refer to heuristics attempting to estimate h^{C*} as *center heuristics*, and to heuristics attempting to estimate $h^{\mathcal{S}*}$ as *star heuristics*, and distinguish them notationally by superscripts "C" respectively "$\mathcal{S}$".

Observe that h^C is a special case of $h^{\mathcal{S}}$: We can compute h^C as $h^{\mathcal{S}}$ in a modified planning task where the cost of all leaf actions is set to 0. h^C keeps track only of which leaf states are reachable, not of the associated cost.

This may lead to qualitatively different decisions, i. e., h^C and $h^{\mathcal{S}}$ may *disagree*. Using a transportation example again, say that there are two alternative kinds of plans, (a) ones that pass the packages through several trucks, loading/unloading every time, vs. (b) ones that make more truck moves but have to load/unload each package only once and thus are better globally. Then h^C will draw search towards plans (a), whereas $h^{\mathcal{S}}$ will draw search towards plans (b).

Search Algorithms

Disregarding optimality, we can run any search algorithm on the decoupled state space, stopping at the first decoupled goal state. For optimal planning, matters are more subtle. One of our methods is formulated on a modified state space

Algorithm DX:
Input: FDR planning task Π, star factoring $\mathcal{F}$
Heuristic search algorithm X
star heuristic $h^{\mathcal{S}}$
Output: A plan for Π, or "failed"
Let $h^{\mathcal{S}}_{G\$} := \begin{cases} h^{\mathcal{S}}(s) & s \in S^{\mathcal{F}} \\ 0 & s = G' \end{cases}$
Run X with $h^{\mathcal{S}}_{G\$}$ on $\Theta^{\mathcal{F}}_{G\$}$
If X found a solution path $\pi = \pi^{\mathcal{F}} \circ \langle s_G \to G' \rangle$
return GlobalPlan($\pi^{\mathcal{F}}$)
else return "failed"

Figure 3: Exploiting any known search algorithm X.

Algorithm ADA^*:
Input: FDR planning task Π, star factoring $\mathcal{F}$
Center heuristic h^C, star heuristic $h^{\mathcal{S}}$
Output: An optimal plan for Π, or "unsolvable"
Let $U := \infty$ /* best known upper bound */
Let $\pi^{\mathcal{F}}_U := \bot$ /* corresponding plan */
Run A* with h^C on $\Theta^{\mathcal{F}}$, with these modifications:
Continue search until the open list is empty
Whenever a goal vertex node $N[s_G]$ is expanded:
If $g(N) + \mathsf{Gprice}(s_G) < U$
let $U := g(N) + \mathsf{Gprice}(s_G)$
let $\pi^{\mathcal{F}}_U :=$ the decoupled plan leading to N
If $\mathsf{Gprice}(s_G) = \mathsf{MINGprice}$
return GlobalPlan($\pi^{\mathcal{F}}_U$) /* early termination */
Whenever a node $N[s]$ is generated, and $U \neq \infty$:
If $g(N) + h^{\mathcal{S}}(s) \geq U$
discard N /* upper-bound pruning */
If $\pi^{\mathcal{F}}_U \neq \bot$ return GlobalPlan($\pi^{\mathcal{F}}_U$) else return "unsolvable"

Figure 4: Anytime search algorithm. Search nodes are notated $N[s]$ where s is the state and N the node itself. MINGprice is the sum, over the leaf factors $F^L \in \mathcal{F}^L$, of optimal plan cost for the *projection* of Π onto F^L.

where the goal pricing functions are explicit. Consider Figure 3. DX ("*Decoupled* X") just runs any search algorithm X on $\Theta^{\mathcal{F}}_{G\$}$, which is defined as $\Theta^{\mathcal{F}}$ with the only difference that from every decoupled goal state $s_G \in S^{\mathcal{F}}_G$, we introduce a new transition to an artificial goal state G'. These transitions have cost $\mathsf{Gprice}(s_G) = \sum_{F_L \in \mathcal{F}} min\ \mathsf{prices}[s_G](s^L)$ with $s_L \in S^L|_{F^L}$ being a leaf goal state. If X is complete, then DX is complete. If X is optimal for admissible heuristics, then DX is optimal for star-admissible heuristics.

Consider now Figure 4. ADA* guides A* by a *center heuristic*, and uses a star heuristic merely for upper-bound pruning. The search is drawn to cheap center paths, disregarding leaf costs, so to guarantee optimality we must exhaust the open list. Without early termination, this would be dominated by DA* because ADA* would then have to expand at least all $N[s]$ where $g(N) + h^{\mathcal{S}}(s)$ is less than optimal solution cost. *With* early termination, that is not so because in the best case we have to exhaust only those

$N[s]$ where $g(N) + h^C(s)$ is less than optimal *center* solution cost. ADA* is complete, and is optimal for center-admissible h^C and star-admissible $h^{\mathcal{S}}$.

Future Work

Having introduced Decoupled Search in the previous chapter, we can now go into future work topics that arise more or less naturally from the pitfalls mentioned above. Besides, it is quite obvious that many search enhancement techniques that have been developed for standard search, can also be adapted to work in the decoupled setting.

A serious drawback of decoupled search, mentioned in the introduction, is that – in principle – the decoupled state space can be infinitely large. This is the case because the pricing functions implicitly remember the center path taken to the decoupled state. Consider a slightly changed version of our illustrative example, that does not require a package to be loaded in a truck to be able to $move$ the truck. With the factoring that has the package as the center and the two trucks as leaves, there are only 5 center states (the different positions of the package). In the initial state, i. e., with the empty center path, the trucks can be located at all three positions with a price of 0 to 2, depending on their initial position. Say we decide to load the package into t_B, using the compliant leaf state $t_B = l_1$ for a price of 2 and to immediately unload the package again, resulting in the decoupled state s_1. Then, the initial center state is identical to that of s_1, but their prices for t_B differ. For the initial state, they are $[2, 1, 0]$ for $[t_B = l_1, t_B = l_2, t_B = l_3]$ and for s_1 they are $[2, 3, 4]$. Consequently, we must consider s_1 to be a new decoupled state. However, it is intuitively obvious, that the initial state "is better" than s_1, more formally: Whatever we can do in s_1, we can at least as cheaply do in the initial state. The question arises what "better" means in this scenario, so how can we define a notion of *dominance* between two decoupled states s_1 and s_2? As already described, the simple notion of dominance employed by (Gnad and Hoffmann 2015) is sufficient to guarantee that the just characterized problem can no longer occur. The dominance check does a pointwise comparison of the prices of the two states and whenever the center states are identical and $\mathsf{prices}[s_1](s_L) \leq \mathsf{prices}[s_2](s_L)$ for all leaf states, s_1 dominates s_2 and we can discard s_2. However, this does not prevent the exponential blow-up of the decoupled state space compared to the standard one, yet. How to further reduce the size of the decoupled state space and possibly upper bound it by the size of the standard state space, is still an open question.

Closely related to this is the idea of not only being able to introduce dominance between decoupled states with the same center, but also when the center states differ. Recent work (Hall *et al.* 2013; Torralba and Hoffmann 2015) has introduced such kind of pruning in the standard state space. It will be a highly interesting research question how to adapt this to the decoupled setting.

In a similar direction goes the pruning of symmetric states, that has been used in planning for several years (e. g. (Pochter *et al.* 2011; Domshlak *et al.* 2012; 2013)). Can we detect symmetries between center states and perform pruning based on some criteria of their corresponding pricing functions? Or is it even possible to detect more global symmetries in the decoupled state space, e. g., considering a standard logistics example. If two packages are completely symmetric, there is no need to handle them separately, but they can be treated as a single package.

In some domains, our current factoring strategies do find factorings with a high number of leaves, but these are very small and the remaining center state space is not much smaller than the whole standard state space. In this case, it makes sense to think of methods further reducing the size of the center state space. One promising approach is partial-order reduction via strong stubborn sets, a technique originally proposed in the context of model checking, that has recently been introduced in planning (Valmari 1989; Wehrle and Helmert 2012; Wehrle *et al.* 2013; Wehrle and Helmert 2014). The inverted setting does occur, too. A factoring resulting in a small center factor where enumerating the state space of few very big leaves leads to a significant runtime overhead. In this case, does it make sense to apply partial-order reduction to the leaves?

Talking about the factoring strategies, so far we only deployed fork-like factorings, though our framework allows for much more general star topologies. Further exploring the space of possible factoring methods is an important task, making decoupled search applicable to a large number of new domains. This not only considers the application to other classical planning domains, but also extends the scope to other search-based areas such as puzzles, or multi-agent planning, but also model checking.

The combination of decoupled search with other known techniques from, e. g., model checking is highly interesting, too. How about a kind of hybrid search that performs standard search in the center part and symbolic search in the leaves? Especially if the leaves have a rather large state space, as, e. g., is very likely if we look at 2-factor star topologies, this promises to solve the problem of the big overhead needed to compute and store the pricing functions.

Overall, there is a great variety of techniques and methods, combining which with decoupled search can enrich the state-of-the-art in planning and related search applications.

Conclusion

Decoupled state space search promises a wide open new research area, exploring which is far beyond the scope of a single dissertation. The initial framework has already been introduced and tightened, so new extensions have a stable basis to build on. The enhancement techniques outlined in the previous chapter are mostly known from standard state space search and can presumably be combined with decoupled search. Being orthogonal to the idea of decoupling – which is exploiting conditional independence between parts of a planning task – a significant additional improvement over the plain variant can be expected when enabling these methods in decoupled search.

References

Eyal Amir and Barbara Engelhardt. Factored planning. In G. Gottlob, editor, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 929–935, Acapulco, Mexico, August 2003. Morgan Kaufmann.

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

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

R. I. Brafman and C. Domshlak. Factored planning: How, when, and when not. In Yolanda Gil and Raymond J. Mooney, editors, *Proceedings of the 21st National Conference of the American Association for Artificial Intelligence (AAAI-06)*, pages 809–814, Boston, Massachusetts, USA, July 2006. AAAI Press.

Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric Hansen, editors, *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pages 28–35. AAAI Press, 2008.

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

Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Enhanced symmetry breaking in cost-optimal planning as forward search. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press, 2012.

Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Symmetry breaking: Satisficing planning and landmark heuristics. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy, 2013. AAAI Press.

Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux. Cost-optimal factored planning: Promises and pitfalls. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 65–72. AAAI Press, 2010.

Daniel Gnad and Jörg Hoffmann. Beating LM-cut with h^{max} (sometimes): Fork-decoupled state space search. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press, 2015.

Daniel Gnad, Jörg Hoffmann, and Carmel Domshlak. From fork decoupling to star-topology decoupling. In Levi Lelis and Roni Stern, editors, *Proceedings of the 8th Annual Symposium on Combinatorial Search (SOCS'15)*. AAAI Press, 2015.

David Hall, Alon Cohen, David Burkett, and Dan Klein. Faster optimal planning with partial-order pruning. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy, 2013. AAAI Press.

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

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

Peter Jonsson and Christer Bäckström. Incremental planning. In *European Workshop on Planning*, 1995.

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

Nir Pochter, Aviv Zohar, and Jeffrey S. Rosenschein. Exploiting problem symmetries in state-based planners. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI-11)*, San Francisco, CA, USA, July 2011. AAAI Press.

Álvaro Torralba and Jörg Hoffmann. Simulation-based admissible dominance pruning. In Qiang Yang, editor, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 1689–1695. AAAI Press/IJCAI, 2015.

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

Martin Wehrle and Malte Helmert. About partial order reduction in planning and computer aided verification. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press, 2012.

Martin Wehrle and Malte Helmert. Efficient stubborn sets: Generalized algorithms and selection strategies. In Steve Chien, Minh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press, 2014.

Martin Wehrle, Malte Helmert, Yusra Alkhazraji, and Robert Mattmüller. The relative pruning power of strong stubborn sets and expansion core. In Daniel Borrajo, Simone Fratini, Subbarao Kambhampati, and Angelo Oddi, editors, *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, Rome, Italy, 2013. AAAI Press.