Decoupled Search for Proving Unsolvability

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

Martin Wehrle
University of Basel
Basel, Switzerland
martin.wehrle@unibas.ch

Introduction

Decoupled State Space Search is a recently introduced method to handle the well-known state space explosion problem (Gnad and Hoffmann 2015; Gnad, Hoffmann, and Domshlak 2015). By exploiting the structure of the problem within the search – as opposed to doing that within a heuristic function guiding the search – the size of the *decoupled state space* can be exponentially smaller than that of the standard state space. Decoupled search achieves that by partitioning the task into several components, called *factors*, trying to identify a *star topology*, with a single *center factor* that interacts with multiple *leaf factors*. By enforcing such a star structure, and thereby simplifying the dependencies between the components, decoupled search has proved to be very efficient and able to compete with other state-of-the-art planners in both satisficing and optimal search. We have also seen good performance of decoupled search in the limited unsolvable benchmarks, available prior to this competition. Whether these results translate to the competition is mainly dependent on the structure of the used domains. Since the currently implemented method to identify factorings is only capable of detecting so-called X-shape profiles, we cannot perform decoupled search in absence of such structure. In such a case, we simply run standard search, instead. As a side remark, this limitation is merely due to the preliminary methods to identify suitable factorings. In general, *every* task has a star topology and can be tackled by decoupled search.

Depending on the particular factoring profile that has been identified, we also enable extensions of decoupled search that have recently been developed, namely partial-order reduction (POR) (Gnad, Wehrle, and Hoffmann 2016) and dominance pruning (Torralba et al. 2016). POR via strong stubborn sets is a technique that is well-known in standard search and originates from the model checking community (Valmari 1989; Alkhazraji et al. 2012; Wehrle and Helmert 2012; 2014). Dominance pruning identifies states that can be safely discarded, without affecting completeness (and optimality). Both of these techniques can only be used if the identified factoring has the form of a fork. We also enable POR, whenever our factoring method does not find a suitable profile.

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 tuple $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ where the *precondition* $\mathsf{pre}(a)$ and *effect* $\mathsf{eff}(a)$ are partial assignments to V.

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. $G \subseteq I[\![\langle a_1, \ldots, a_n\rangle]\!]$. For the task of proving a task unsolvable, we are only interested in the existence of a plan that transforms the initial state I to a goal state s_G, with $s_G[v] = G[v]$ for all $v \in \mathcal{V}(G)$. We consequently ignore action costs in the following.

To identify the required structure for factoring the variables, we need the notion of the *causal graph* (e. g. (Knoblock 1994; Jonsson and Bäckström 1995; Brafman and Domshlak 2003; Helmert 2006)). The causal graph of a planning task captures state variable dependencies. 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 $(v \to v')$, 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 con-

nected (this is wlog: else, the task can be equivalently split into several independent tasks).

Decoupled Search

We run decoupled search like introduced by Gnad, Hoffmann, and Domshlak (2015), with the same factoring strategy and search settings, i. e., optimized for satisficing search. Since there is no difference in the main algorithm, we only give a brief summary, here.

Prior to search, the factoring of the input task Π is performed, by analyzing its causal graph. Denote by $\mathcal{F}^{\text{SCC}}$ the factoring whose factors are the strongly connected components (SCC) of *CG*. The *interaction graph* $IG(\mathcal{F})$ of a factoring $\mathcal{F}$ 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 *CG*. The actual factoring works as follows: In a first step, each leaf in $\mathcal{F}^{\text{SCC}}$ will be assigned to a single leaf factor F^L. If the causal graph is not strongly connected, i. e., at least one such leaf exists, this results in a fork factoring, where – denoting by $F^{C'}$ the remaining components – all transitions in $IG(\mathcal{F}^{\text{SCC}})$ are of the form $(F^{C'} \to F^L)$. In a second step, each root from the sub-graph of $IG(\mathcal{F}^{\text{SCC}})$ that only contains the components in $F^{C'}$ is also assigned to a new leaf factor. By F^C we denote the remaining components that have not been assigned to a leaf. Finally, all leaves F^{L_2} detected in the second step that introduce transitions in $IG(\mathcal{F}^{\text{SCC}})$ of the form $(F^{L_2} \to F^{L_1})$ will be put back into F^C, to prevent dependencies across leaf factors. If leaves have been detected in both steps and at least one of those from step 2 has not been removed, this results in X-shape factoring with "inverted-fork" leaves that provide preconditions for the center, and "fork" leaves, that only have preconditions on the center and themselves. If only in the second step leaves have been added, this results in a pure inverted-fork factoring.

Given a factoring $\mathcal{F}$ with center factor F^C and leaves $F^L \in \mathcal{F}^L$, decoupled search is performed as follows:

The search will only branch over center actions, i. e., those actions affecting a variable in F^C. Along such a path of center actions π^C, for each leaf factor F^L, the search maintains a set of leaf paths, i. e., actions only affecting variables of F^L, that *comply* with π^C. Intuitively, for a leaf path π^L to comply with a center path, it must be possible to embed π^L into π^C such that the F^L-preconditions of all center actions are provided by π^L at the respective points in π^C, and the F^C preconditions of all leaf actions are provided by π^C.

A decoupled state corresponds to an end state of such a center action sequence. The main advantage over standard search originates from a decoupled state being able to represent exponentially many explicit states, thereby getting rid of having to enumerate all of them. A decoupled state can "contain" many explicit states, because by instantiating the center with a center action sequence, the leaf factors are mutually independent. Thus, the more leaves in the factoring, the more explicit states can potentially be represented by a single decoupled state.

Decoupled strong stubborn sets

In addition to the plain decoupled search variant outlined above, we enable decoupled strong stubborn sets (DSSS), when the factoring method results in a fork topology. The usage of this technique is identical to what has been introduced in Gnad, Wehrle, and Hoffmann (2016), so we don't give the formal details, here. DSSS are a straightforward extension of POR to the decoupled search setting, where some care must be taken due to the specific structure of the decoupled state space, especially the distinction between center and leaf actions. Like in the standard state space, it removes transitions that will lead to different permutations of action sequences leading to the same outcome state. The only minor difference to the original implementation is a "safety belt", that disables DSSS if after the first 1000 expansions, not a single transition has been removed.

Decoupled dominance pruning

Another decoupled search extension that has only recently been introduced is dominance pruning (Torralba et al. 2016), where decoupled states that are dominated by other – already visited – states can be safely discarded. We only deploy a very lightweight pruning method, namely *frontier* pruning. The plain decoupled search variant performs a duplicate checking that can already detect certain forms of dominance, in particular if two decoupled states have the same center state and all leaf states reachable in one state are (at most as costly) also reachable in the other. Frontier pruning improves this by only comparing a subset of the reached leaf states, those that can possibly make so far unreached leaf states available. It has originally been developed for optimal planning, but can be easily adapted to become more efficient, when optimal solutions do not matter, by replacing the real cost of reaching a leaf state by 0, if a state has been reached at any cost.

Additionally, we also employ a leaf simulation, originally proposed by Torralba and Kissmann (2015), to remove superfluous leaf states and leaf actions, discovering transitions that can be replaced by other transitions, then running a reachability check on the leaf state space. In some domains, this can tremendously reduce the size of the leaf state spaces.

Implementation

Decoupled Search has been implemented as an extension of the Fast Downward (FD) planning system (Helmert 2006). By changing the low-level state representation, many of FD's built-in algorithms and functionality can be used with only minor adaptations. Of particular interest for the task of proving unsolvability are the A* algorithm, the h^{max} heuristic (Bonet and Geffner 2001) for dead-end pruning, and partial-order reduction via strong stubborn sets. On top of the standard FD preprocessor, we perform a relevance analysis based on h^2, in order to eliminate actions and simplify the planning task prior to the search (Alcázar and Torralba 2015). In some domains, this relevance analysis is even powerful enough to detect a task unsolvable without actually having to start the search.

All our search variants run A* using h^{max}. The actual search configuration depends on the identified factoring $\mathcal{F}$ as follows:

(i) $|\mathcal{F}| \leq 2$ (at most 1 leaf factor): Run standard search using strong stubborn sets.

(ii) $|\mathcal{F}| > 2$ (at least 2 leaf factor) and $\mathcal{F}$ is a fork factoring: Run decoupled search using decoupled strong stubborn sets, frontier dominance pruning, and leaf simulation.

(iii) $|\mathcal{F}| > 2$ (at least 2 leaf factor) and $\mathcal{F}$ is not a fork factoring: Run decoupled search without extensions.

The combination of decoupled strong stubborn sets *and* dominance pruning has not been formally described, before. We are not going into this, either, but rather give the intuition of why this still results in a complete search algorithm. Given a set of actions applicable in a state, decoupled strong stubborn sets prune the subset of these actions, that start different permutations of actions leading to the same outcome state. By guaranteeing that one of these permutations applicable in the current state will not be pruned, search using strong stubborn sets remains complete (and optimal).

In contrast to that, dominance pruning removes a state s that is dominated by another decoupled state t. By analyzing the structure of the leaf factors and comparing only the relevant leaf states of s to those of t, it can detect that all states that are reachable from s are also (at most as costly) reachable from t.

Putting things together, decoupled strong stubborn sets and dominance pruning are orthogonal methods to reduce the size of the state space – one removes transitions that correspond to redundant permutations of action sequences, the other removes states that cannot reach anything that could not be reached before. Consequently, it is safe to combine both, resulting in a state space that can be significantly smaller than when only using one of the techniques.

The case distinction outlined above allows a flexible adaptation to the given input problem, and since computing the factoring in most tasks finishes within split seconds, there is (almost) no computational overhead to determine which search variant to use.

Acknowledgments. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/6-1, "Star-Topology Decoupled State Space Search".

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. 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.

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

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

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

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

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.

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

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

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

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

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

Torralba, Á., and Kissmann, P. 2015. Focusing on what really matters: Irrelevance pruning in merge-and-shrink. In Lelis, L., and Stern, R., eds., *Proceedings of the 8th Annual Symposium on Combinatorial Search (SOCS'15)*, 122–130. AAAI Press.

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

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

Wehrle, M., and Helmert, M. 2012. About partial order reduction in planning and computer aided verification. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press.

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