Advanced Factoring Strategies for Decoupled Search using Linear Programming

Frederik Schmitt, Daniel Gnad, Jörg Hoffmann
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
s8fkschm@stud.uni-saarland.de, {gnad,hoffmann}@cs.uni-saarland.de

Abstract

Star-topology decoupled state space search decomposes a planning task and searches over the component state spaces instead of multiplying the state variables. This can lead to an exponential reduction of the search effort. To do so, in a preprocess before the search, the given planning task is partitioned into factors, such that the interaction between these factors takes the form of a star topology. Prior work has identified several ways to automatically decompose planning tasks, however, was not able to release the full potential of decoupled search. We try to close this gap by introducing an integer linear programming formulation of the factoring process, allowing us to explicitly specify the properties that a factoring should have. We prove that our approach returns the factoring that maximizes the number of factors, if this is the objective, and employ two other properties to assess the quality of a factoring. Our experimental evaluation shows that this leads to superior performance and substantially increases the applicability of decoupled search.

Introduction

In classical planning, star-topology decoupled state space search, or decoupled search for short, has been introduced as a reduction method to compactly represent huge transition systems (Gnad and Hoffmann 2018). To do so, the variables of a given planning task are partitioned into a set of disjoint factors, such that the interaction between these factors takes the form of a star topology. This factoring process is crucial for the performance of decoupled search, highly influencing its reduction power. It results in a *factoring* of the state variables, such that there exists a *center factor* C and a set of *leaf factors* $\mathcal{L}$, where the interaction between any two leaves also has to involve the center.

Prior work has identified factoring strategies that are based on the causal graph of a task (e. g., Knoblock (1994), Jonsson and Bäckström (1995), Brafman and Domshlak (2003), Helmert (2006)), by analyzing its strongly connected components if the causal graph is only weakly connected (Gnad and Hoffmann 2015; Gnad, Hoffmann, and Domshlak 2015), or computing a maximum independent set if it is strongly connected (Gnad, Poser, and Hoffmann 2017). The latter work also proposed a greedy strategy that puts variables densely connected in the causal graph (with many incident arcs) into the center, making each weakly connected component in the remainder a leaf factor.

All existing factoring methods have in common that they look for so-called *strict-star* factorings, where no interaction between any two leaves is allowed. Our method is not restricted to strict stars, but able to identify *general-star* factorings, where interaction between leaves is allowed as long as the center is affected at the same time. We do so by formulating the factoring process as an integer linear program (ILP). Integer linear programming is a well-known mathematical optimization problem. It has gained increased attention in classical planning due to its descriptive power to express complex properties (e. g., Bylander (1997), Bonet and van den Briel (2014), Pommerening et al. (2014)).

We encode both types of factorings as constraints of an ILP with a decision variable x_{L_i} for each of a set of identified *potential* leaf factors. The constraints ensure that the resulting factoring is valid. We formulate different objective functions that we want to maximize, e. g., the number of leaf factors. We prove that our ILP encoding guarantees to return the strict/general-star factoring with the maximum number of *mobile* leaf factors, if such a factoring exists, where we call a leaf L mobile if there exists an action with an effect only on the variables of L. Following Gnad, Poser, and Hoffmann (2017), we also encode the *mobility* and *flexibility* of a factoring in the objective function of the ILP, and show that maximizing these can be beneficial over maximizing the number of mobile leaves.

Our evaluation shows a substantial increase in the number of planning instances that can be tackled with our factoring strategies, and that we can even improve in instances where existing strategies already performed well.

For space reasons, we only state proof sketches. Full proofs (and a running example) are available in an online TR (Schmitt, Gnad, and Hoffmann 2019).

Background

We use the finite-domain representation (FDR) of planning (Bäckström and Nebel 1995; Helmert 2006), where a *planning task* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$. Here, $\mathcal{V}$ is a set of *variables*, each $v \in \mathcal{V}$ associated with a finite domain $\mathcal{D}(v)$. We identify (partial) variable assignments with sets of variable/value pairs. A *state* is a complete assignment to $\mathcal{V}$. I is the *initial state* of Π, and G is its *goal*, a partial assignment to $\mathcal{V}$. For a partial assignment p, we denote by $vars(p) \subseteq \mathcal{V}$ the subset of variables on which p is defined. For $V \subseteq vars(p)$, by $p[V]$ we denote the assignment to V made by p. We say that a (partial) assignment p *satisfies* a condition q, denoted $p \models q$, if $vars(q) \subseteq vars(p)$, and $p[v] = q[v]$ for all $v \in vars(q)$. $\mathcal{A}$ is a finite set of *actions*, each a triple $\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle$ of *precondition*, *effect*, and *cost*, where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are partial assignments to $\mathcal{V}$, and $\mathsf{cost}(a) \in \mathbb{R}^{0+}$. An action a is *applicable* in a state s if $s \models \mathsf{pre}(a)$. Applying a in s changes the value of all $v \in vars(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$, and leaves s unchanged elsewhere. The outcome is denoted by $s[\![a]\!]$. A *plan* for Π is an action sequence π iteratively applicable in I and ending in a state s_G s.t. $s_G \models G$. The plan is *optimal* if its summed-up cost, denoted $\mathsf{cost}(\pi)$, is minimal among all plans.

Our factoring strategies are based on the *causal graph*, which captures the structure of a task in terms of pairwise state-variable dependencies (e. g., Knoblock (1994), Jonsson and Bäckström (1995), Brafman and Domshlak (2003), Helmert (2006)). The causal graph CG_Π of a planning task Π is a directed graph whose vertices are the variables $\mathcal{V}$. The graph has an arc (v, v') if $v \neq v'$, and there exists an action $a \in \mathcal{A}$ such that $v \in vars(\mathsf{pre}(a)) \cup vars(\mathsf{eff}(a))$ and $v' \in vars(\mathsf{eff}(a))$. We assume that CG_Π is weakly connected. Otherwise, we can separate each weakly connected component into a sub-task and solve it independently.

Integer linear programming (ILP) is a well-known mathematical optimization problem. A linear program is described as a set of linear constraints $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ over a set of variables x_j, and a linear objective function $\sum_{j=1}^{n} o_j x_j$ over the variables that should be maximized.

Decoupled Search

Due to space restrictions, we only give the minimal required background to decoupled search. We refer the reader to Gnad and Hoffmann (2018) for full details.

The most relevant part for us is the factoring process, where, prior to the actual search, the planning task Π is decomposed by partitioning its state variables $\mathcal{V}$ into non-empty subsets, the *factoring* $\mathcal{F}$. A factoring is a *general-star* factoring, if there exists a *center factor* $C \in \mathcal{F}$, such that, denoting by $\mathcal{L} := \mathcal{F} \setminus \{C\}$ the *leaf factors*, for all actions $a \in \mathcal{A}$ where $vars(\mathsf{eff}(a)) \cap C = \emptyset$, there exists a leaf $L \in \mathcal{L}$ such that $vars(\mathsf{eff}(a)) \subseteq L$ and $vars(\mathsf{pre}(a)) \subseteq L \cup C$. We call actions that affect (with an effect on) C *center actions* $\mathcal{A}^C$, and actions affecting an $L \in \mathcal{L}$ *leaf actions* $\mathcal{A}^L$. In words, there is no restriction for center actions in general-star factorings; leaf actions of a leaf L without center effect, the *leaf-only* actions $\mathcal{A}^L \setminus \mathcal{A}^C$, are restricted to effects on L only, and preconditions on C and L. A factoring $\mathcal{F}$ is a *strict-star* factoring, if no two leaves $L \neq L' \in \mathcal{L}$, are connected via an edge (v, v') in CG_Π, where $v \in L$ and $v' \in L'$.

Leaf-only actions are important for us, since their number quantifies the possible state space size reduction of decoupled search compared to standard search. This is because decoupled search only branches over the center actions, maintaining the set of leaf states (assignments to an $L \in \mathcal{L}$) reachable via leaf-only actions separately. The latter is possible because the leaves are *conditionally independent*, the only interaction between leaves is via the center. Due to this independence, no multiplication across leaves is necessary, the possible reduction is exponential in the number of leaves, and linear, for each leaf, in the number of its leaf-only actions. Therefore, our factoring methods aim at maximizing the number of leaf-only actions. Like Gnad, Poser, and Hoffmann (2017), we use the *mobility* and *flexibility* of leaves to capture the number of leaf-only actions of a factoring. The mobility of a leaf $L \in \mathcal{L}$ is the number of its leaf-only actions $|\mathcal{A}^L \setminus \mathcal{A}^C|$; its flexibility is the ratio of leaf-only to all leaf actions of a leaf, $|\mathcal{A}^L \setminus \mathcal{A}^C| / |\mathcal{A}^L|$. With a mobility/flexibility of 0, a leaf does not contribute to the reduction, and its variables can be moved into the center. We call such leaves *frozen*. Leaves that are not frozen are called *mobile* and a factoring $\mathcal{F}$ is mobile if all its leaves are mobile.

ILP Formulation of the Factoring Process

When we started to work on this paper, we experimented with an ILP encoding of factorings that searches over all possible combinations of the state variables across any number of leaf factors. While this is an interesting concept in theory, possibly resulting in factorings that are different from the ones that we introduce next, it is infeasible to solve the generated ILPs for non-trivial planning instances, due to the high number of constraints. Furthermore, the method never resulted in better factorings when given reasonable runtime limits. Therefore, we herein only introduce a more efficient encoding, that builds on the observation that a leaf is mobile iff it subsumes the variables of at least one *effect schema*. This avoids the complete search of the naive approach while preserving the guarantee that we can construct the factoring that maximizes the number of mobile leaf factors.

An effect schema $E \subseteq \mathcal{V}$ is a subset of the variables of Π, such that there exists an action $a \in \mathcal{A}$ with $vars(\mathsf{eff}(a)) = E$. We denote the set of all effect schemas of a task by $\mathbf{ES}_\Pi$. Observe that, if we want to obtain a *mobile* factoring $\mathcal{F}$, then every leaf $L \in \mathcal{L}$ must be the superset of at least one effect schema $E \in \mathbf{ES}_\Pi$. Every leaf L for which there does not exist an effect schema $E \subseteq L$ is necessarily frozen. Based on this observation, we consider each effect schema $E \in \mathbf{ES}^*_\Pi := \mathbf{ES}_\Pi \setminus \{\mathcal{V}\}$ as a *potential leaf*, and construct a graph with nodes $\mathbf{ES}^*_\Pi$, such that any independent set of the graph forms a proper factoring. Since these graphs are different for strict and general-star factorings, we dedicate a separate subsection to each of the two.

Strict-star factorings

The *potential strict-leaf graph* $\mathbf{PSLG}_\Pi(\mathcal{S})$ is an undirected graph with vertices $\mathcal{S} \subseteq \mathcal{P}(\mathcal{V}) \setminus \{\emptyset, \mathcal{V}\}$, where $\mathcal{P}(\mathcal{V})$ denotes the powerset of $\mathcal{V}$, and edges (i) (S, S') if $S \neq S'$ and $S \cap S' \neq \emptyset$, and (ii) (S, S') if $S \neq S'$ and there exist $v \in S$ and $v' \in S'$, such that (v, v') is an edge in CG_Π. So, $\mathbf{PSLG}_\Pi(\mathcal{S})$ has an edge between (i) every pair of overlapping variable sets, and (ii) sets that are connected via their variables in the causal graph. Note that this exactly captures the requirements of a factoring, such that two connected sets cannot both be leaves in a strict-star factoring.

	B	F	IF	IA	MIS	SL	SF	SM	GL	GF	GM	Cov.	Abs.	t/o	m/o
Base	-	1	1	6	4	5	6	5	6	**8**	6	942	0	0	0
F	**5**	-	**5**	**7**	**5**	**7**	**6**	5	**8**	**8**	**6**	**975**	1367	0	0
IF	**3**	4	-	**9**	**7**	**7**	**8**	6	**8**	**10**	**8**	956	1364	0	0
IA	**8**	4	6	-	**5**	7	6	4	**8**	**9**	6	950	805	0	0
MIS	**5**	2	4	3	-	4	5	2	**7**	**8**	5	946	1097	22	0
SL	**9**	6	6	**8**	**7**	-	**3**	1	**6**	**9**	**7**	956	459	85	16
SF	**9**	**6**	7	**9**	**6**	**3**	-	0	**8**	**7**	**6**	960	459	86	15
SM	**10**	**7**	**7**	**11**	**8**	**5**	**3**	-	**10**	**9**	**6**	968	459	89	13
GL	**8**	6	5	7	**7**	4	4	3	-	**4**	4	934	359	188	9
GF	**8**	6	6	7	5	5	3	3	**4**	-	3	941	359	190	9
GM	**8**	**6**	6	**8**	**6**	6	4	2	**7**	**5**	-	948	359	189	9

Figure 1: Coverage comparison in optimal planning when using the LM-cut heuristic. "Cov." is the total coverage, "Abs." the number of abstained instances, and "t/o" ("m/o") the number of instances where the factoring timed out (ran out of memory). An entry in row A, column B shows the number of domains in which method A has strictly higher coverage than method B; bold facing for each pair (A, B), (B, A) the maximal entry.

	B	F	IF	IA	MIS	SL	SF	SM	GL	GF	GM	Cov.	Abs.	t/o	m/o
Base	-	2	1	7	2	8	9	9	12	11	11	1320	0	0	0
F	**9**	-	5	**9**	**6**	9	9	**9**	**10**	10	10	1350	1339	0	0
IF	**11**	**11**	-	**14**	**11**	**13**	**12**	**13**	**15**	**13**	**13**	**1432**	1352	0	0
IA	**10**	5	3	-	**5**	8	8	10	**13**	11	12	1349	859	0	0
MIS	**6**	3	3	**5**	-	8	8	9	**12**	11	11	1345	1079	30	0
SL	**13**	10	7	**12**	**11**	-	**6**	**4**	**7**	7	8	1378	390	149	7
SF	**13**	10	6	**10**	**10**	5	-	**6**	**9**	5	**9**	1383	390	150	6
SM	**12**	**9**	7	**11**	**10**	**4**	5	-	7	6	7	1388	390	150	6
GL	**14**	9	8	**13**	11	6	7	**8**	-	4	**6**	1351	317	216	13
GF	**15**	**11**	8	**13**	**12**	**9**	**6**	**8**	**7**	-	**7**	1371	317	216	13
GM	**14**	**11**	9	**14**	**12**	**10**	**9**	**8**	**6**	4	-	1376	317	217	13

Figure 2: Coverage comparison when using the h^{FF} heuristic. For explanations, see caption of Figure 1 and text.

Lemma 1 *Let Π be a planning task and let $\mathcal{S} = \mathcal{P}(\mathcal{V}) \setminus \{\emptyset, \mathcal{V}\}$, where $\mathcal{P}(\mathcal{V})$ is the powerset of $\mathcal{V}$. Then $\mathcal{F} = \{L_1, \dots, L_n, C\}$ is a strict-star factoring for Π, iff $I = \{L_1, \dots, L_n\}$ forms an independent set of $\mathbf{PSLG}_\Pi(\mathcal{S})$, where $C = \mathcal{V} \setminus \bigcup_{i=1}^{n} L_i$.*

Proof Sketch: The potential strict-leaf graph $\mathbf{PSLG}_\Pi(\mathcal{S})$ encodes exactly the conditions that a strict-star factoring $\mathcal{F}$ has to satisfy. Namely, $\mathcal{F}$ has to be a partitioning of $\mathcal{V}$, which is guaranteed by type (i) edges in the graph, and for any pair of leaves $L_1 \neq L_2 \in I$, there must not be a causal-graph connection between contained variables, which is guaranteed by the type (ii) edges. □

With Lemma 1, and the observation that every leaf that subsumes an effect schema is mobile, we can construct *mobile* strict-star factorings from the independent sets of $\mathbf{PSLG}_\Pi(\mathbf{ES}^*_\Pi)$, and vice versa.

Theorem 1 *Let Π be a planning task and let $\mathbf{ES}^*_\Pi$ be the set of its potential leaves. Then from an independent set of size k of $\mathbf{PSLG}_\Pi(\mathbf{ES}^*_\Pi)$ we can construct a mobile strict-star factoring $\mathcal{F}$ with k leaves, and vice versa.*

It follows that from a maximum independent set of $\mathbf{PSLG}_\Pi(\mathbf{ES}^*_\Pi)$ we can construct a strict-star factoring $\mathcal{F}$ with the maximum number of mobile leaves.

General-star factorings

The *potential general-leaf hypergraph* $\mathbf{PGLG}_\Pi(\mathcal{S})$ is a hypergraph with vertices $\mathcal{S} \subseteq \mathcal{P}(\mathcal{V}) \setminus \{\emptyset, \mathcal{V}\}$, and edges (i) $\{S, S'\}$ if $S \cap S' \neq \emptyset$, (ii) $\{S, S'\}$ if there exist $v \in S$, and $a \in \mathcal{A}$, such that $v \in \mathit{vars}(\mathsf{pre}(a))$, and $\mathit{vars}(\mathsf{eff}(a)) \subseteq S'$, and (iii) $\{S_1, \dots, S_n\}$ if $n > 1$ and there exists an action $a \in \mathcal{A}$ such that $\mathit{vars}(\mathsf{eff}(a)) \subseteq S_1 \cup \dots \cup S_n$ and $\forall S_i : \mathit{vars}(\mathsf{eff}(a)) \cap S_i \neq \emptyset \wedge \mathit{vars}(\mathsf{eff}(a)) \not\subseteq S_i$. $\mathbf{PGLG}_\Pi(\mathcal{S})$ has an edge between (i) every pair of overlapping variable sets as in the strict-star case. Type (ii) edges prevent precondition-effect dependencies between two leaves caused by leaf-only actions $a \in \mathcal{A}^L \setminus \mathcal{A}^C$. Condition (iii) ensures that no action $a \in \mathcal{A}$ in the obtained factoring affects more than one leaf without affecting the center at the same time. At least one of $S_1, \dots, S_n$ cannot become a leaf, since otherwise a affects only leaf factors. Note that the last part of the condition ($\mathit{vars}(\mathsf{eff}(a)) \not\subseteq S_i$) only prevents hyperedges between sets that are connected by type (i) edges, anyway.

As in the strict-star case, we can prove that from an independent set of size k of $\mathbf{PGLG}_\Pi(\mathbf{ES}^*_\Pi)$ we can construct a mobile general-star factoring $\mathcal{F}$ with k leaves. The proof idea is similar to that of Theorem 1.

Theorem 2 *Let Π be a planning task and let $\mathbf{ES}^*_\Pi$ be the set of its potential leaves. Then from an independent set of size k of $\mathbf{PGLG}_\Pi(\mathbf{ES}^*_\Pi)$ we can construct a mobile general-star factoring $\mathcal{F}$ with k leaves, and vice versa.*

ILP construction & optimization criteria

We construct ILPs that compute independent sets on the aforementioned graphs, maximizing an objective function that encodes the desired property. For each potential leaf $L \in \mathbf{ES}^*_\Pi$, we have a binary variable x_L that encodes if L is a leaf ($x_L = 1$) or not. We add a constraint for every (hyper-)edge in the potential-leaf graphs $\mathcal{G}$:

$$x_{L_1} + \dots + x_{L_k} \leq k - 1 \quad \text{for } \{L_1, \dots, L_k\} \in \mathcal{G}$$

For every potential leaf, we specify an *objective value* o_L, which is 1 if we aim at maximizing the number of mobile leaves, or the mobility or flexibility of the leaf. Then $\sum_{L \in \mathbf{ES}^*_\Pi} o_L \cdot x_L$ is the objective function of the ILP, maximizing the sum of the respective measures in the factoring.

Experiments

We implemented our new factoring strategies in Fast Downward (FD) (Helmert 2006), extending the decoupled search implementation of Gnad and Hoffmann (2018). To solve the ILPs, we use the IBM CPLEX solver[1]. We conduct experiments in optimal and satisficing planning, using all IPC STRIPS benchmarks (1998 – 2018) with a total of 1819 instances in each track, distributed across 66/65 domains for

[1]https://www.ibm.com/analytics/cplex-optimizer

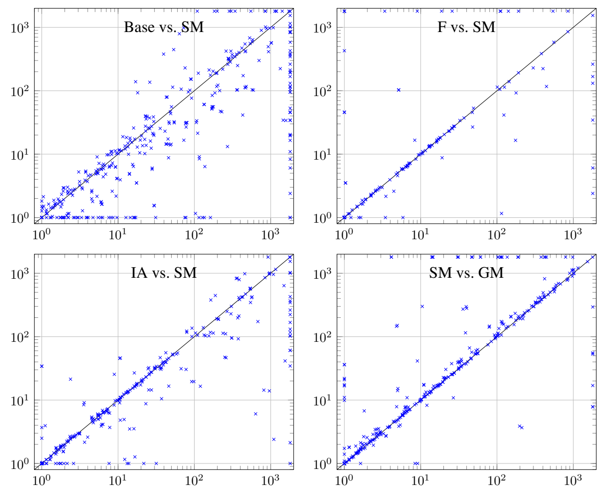

Figure 3: Scatter plots showing the runtime of A vs. B, with A on the x-axis and B on the y-axis, in optimal planning.

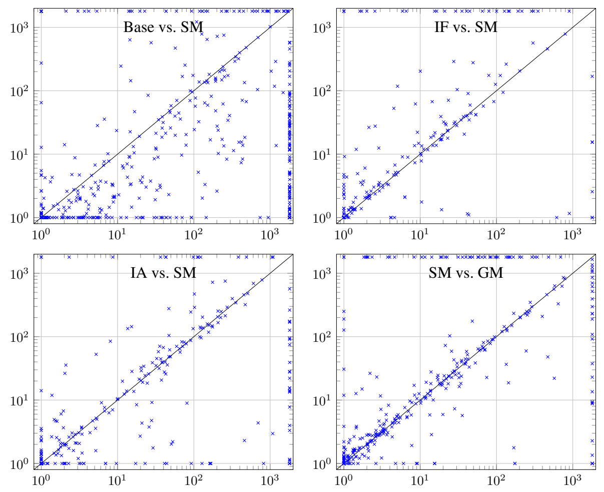

Figure 4: Runtime scatter plots for satisficing planning.

the optimal/satisficing suite. In the optimal planning setting, all configurations run A* search using the LM-cut heuristic (Helmert and Domshlak 2009); in satisficing planning, we run greedy best-first search with the h^{FF} heuristic (Hoffmann and Nebel 2001). The experiments were performed on a cluster of Intel E5-2660 machines running at 2.20 GHz, with time (memory) cut-offs of 30 minutes (4 GB).

We compare to standard search (**Base**), and all factoring strategies from Gnad, Poser, and Hoffmann (2017), namely fork (**F**), inverted-fork (**IF**), incident-arc (**IA**), and maximum independent set (**MIS**). We use our new strict-star (**S***) and general-star (**G***) encodings, maximizing the number of mobile leaves (***L**), flexibility (***F**), or mobility (***M**).

We adopt the *abstain* mechanism of prior work on decoupled search, abstaining from solving a task if the factoring has less than two leaf factors. Moreover, we implemented a *timeout* of 30s for the factoring process. We take the best factoring found up to this point (if one exists), running CPLEX in anytime mode. Not imposing the limit leads to not abstaining on 113 additional instances, and 5 instances with slightly higher objective value, across all benchmarks for GM, for which this happens most often. Yet, abstaining on the former instances and running Base after the timeout (cf. next paragraph) leads to superior coverage ($+14/+1$ in the satisficing/optimal setting). We conclude that the factorings found on instances with tightly coupled graphs – those where it is hard to find an independent set – do not lead to significant performance gains of decoupled search over standard search. Our slowest factoring (GM) terminates within 1s in 74% and within 30s in 86% of all tasks.

Figures 1 and 2 compare the coverage of our new factorings to the existing methods and standard search. For the decoupled search methods, an instance is considered solved if the factoring succeeds and decoupled search solves the task, or if the factoring fails but the sum of factoring time and runtime of the baseline in the task is smaller than 30min (we can still run standard search if no factoring is found). Our new factoring methods tackle significantly more instances, and perform better than both standard search and the existing methods in many cases. Strict-star factorings are always better (in total coverage) than general-star factorings, although the latter abstain less. Computing a general-star factoring is computationally more costly, causing the methods to time out more often than the strict-star configurations. Interestingly, the "*M" variants consistently outperform "*L" and "*F", so just having more leaves does not seem to be beneficial in general. On our benchmark suites, Complementary2 (Franco, Lelis, and Barley 2018) solves 1075 instances; LAMA (Richter and Westphal 2010) solves 1606.

Figures 3 and 4 directly compare the runtime (FD's "search-time") of some pairs of configurations. The plots include all instances where neither method abstains (all for Base). Strict-star factorings also seem to perform better than general-stars in terms of runtime. Except IF in satisficing planning, which performs clearly better than SM (except in a few instances), there is quite a large number of instances where we see a significant speed-up compared to the existing methods. SM is also often substantially faster than Base.

Conclusion

In this work, we introduced new factoring methods for decoupled search that are based on integer linear programming (ILP). The encoding builds on the effect schemas of a planning task, making each of them a potential leaf factor. We formulate the conditions required for strict and general-star factorings as a graph that connects two potential leaves if they cannot both become actual leaf factors of a factoring. We prove that the constructed factorings maximize the number of mobile leaf factors, if this is the objective function of the ILP. Our experiments show that the new methods can tackle significantly more planning instances, lead to an increased performance compared to standard search, and even compared to previous factoring methods in many instances.

Acknowledgments

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

References

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

Bonet, B., and van den Briel, M. 2014. Flow-based heuristics for optimal planning: Landmarks and merges. In Chien, S.; Do, M.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, 47–55. AAAI Press.

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

Bylander, T. 1997. A linear programming heursitic for optimal planning. In Kuipers, B. J., and Webber, B., eds., *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence (AAAI'97)*, 694–699. Portland, OR: MIT Press.

Franco, S.; Lelis, L. H.; and Barley, M. 2018. The complementary2 planner in the IPC 2018. In *IPC 2018 planner abstracts*.

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)*, 88–96. AAAI Press.

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

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)*, 53–61. AAAI Press.

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

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

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.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based heuristics for cost-optimal planning. In Chien, S.; Do, M.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, 226–234. AAAI Press.

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

Schmitt, F.; Gnad, D.; and Hoffmann, J. 2019. Advanced factoring strategies for decoupled search using linear programming – technical report. Technical report, Saarland University. Available at *http://fai.cs.uni-saarland.de/hoffmann/papers/icaps19b-tr.pdf*.