SaarPlan: Combining Saarland's Greatest Planning Techniques

Maximilian Fickert and **Daniel Gnad** and **Patrick Speicher** and **Jörg Hoffmann**
Saarland Informatics Campus
Saarland University
Saarbrücken, Germany
{fickert,gnad,speicher,hoffmann}@cs.uni-saarland.de

Abstract

Saarland is the smallest – yet arguably one of the most beautiful – state in Germany. And it has a lot to offer! From rich nature, over its industrial heritage, really smart people, to powerful planning techniques. SaarPlan combines the best of these planning techniques to a performant portfolio, making it the best planner in Saarland.

Introduction

Among many other things, Saarland offers a wide range of powerful planning techniques. SaarPlan combines the best of these techniques into a portfolio planner. Since in Saarland we don't care too much about optimality, but rather about getting things done, SaarPlan participates in the satisficing, agile, and bounded-cost tracks of the competition.

Some of the ingredients of SaarPlan are also used in *DecStar* (Gnad, Shleyfman, and Hoffmann 2018), and the *OLCFF* planner (Fickert and Hoffmann 2018). From DecStar, SaarPlan takes the *Star-topology decoupled search* part, trying to decompose a given planning task, if possible. In case a good problem decomposition was detected, decoupled search typically performs very well. Finding a decomposition is fast, it succeeds (or fails) quickly, so not much time is lost in the latter case. If it fails, SaarPlan tries its best with *Grey planning*, an enhancement of the red-black planning method used by the Mercury planner (Katz and Hoffmann 2014). The grey planning component of SaarPlan only considers the initial state, generates a semi-delete-relaxed plan and attempts to repair it into a real plan. Again, this process terminates quickly. If these two techniques do not manage to solve the planning task, SaarPlan uses another semi-delete relaxation method, the online-refined $h^{C\text{FF}}$ heuristic. It uses different search methods with this heuristic: an enforced hill-climbing with additional *novelty pruning*, which is followed by a greedy best-first search (GBFS) using the $h^{C\text{FF}}$ heuristic, without pruning. The GBFS part, which is very LAMA-like (Richter and Westphal 2010) – replacing the fully delete-relaxed heuristic with $h^{C\text{FF}}$ – completes SaarPlan, making it Saarland's best planner.

Big things always start in the small, ("Großes entsteht immer im Kleinen!") as we say in Saarland. Does this also hold for planning? Do great planners come from the small Saarland? The competition will answer this question.

Decoupled Search

We perform decoupled search like introduced by Gnad and Hoffmann (2018), in its integration in the Fast Downward planning system (Helmert 2006). We use the improved *fork* and *inverted-fork*, as well as the *incident-arcs* factoring methods from Gnad, Poser, and Hoffmann (2017). The outcome of the factoring process is a partitioning $\mathcal{F}$ of the variables of the planning task Π, such that $|\mathcal{F}| > 1$ and there exists $F^C \in \mathcal{F}$ such that, for every action a where $\mathcal{V}(\mathsf{eff}(a)) \cap F^C = \emptyset$, there exists $F \in \mathcal{F}$ with $\mathcal{V}(\mathsf{eff}(a)) \subseteq F$ and $\mathcal{V}(pre(a)) \subseteq F \cup F^C$. We then call $\mathcal{F}$ a *star factoring*, with *center factor* F^C and *leaf factors* $\mathcal{F}^L := \mathcal{F} \setminus \{F^C\}$.

Given a factoring $\mathcal{F}$, decoupled search is performed as follows: The search will only branch over center actions, i. e., those actions affecting (with an effect on) 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 π^C, it must be possible to embed π^L into π^C into an overall action sequence π, such that π is a valid path in the projection of the planning task Π onto $F^C \cup F^L$. 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, avoiding their enumeration. A decoupled state can "contain" many explicit states, because by instantiating the center with a center action sequence, the leaf factors are conditionally independent. Thus, the more leaves in the factoring, the more explicit states can potentially be represented by a single decoupled state.

We will next describe a couple of extensions that have been developed for decoupled search and that we use in some of our configurations.

Symmetry Breaking in Decoupled Search

Symmetry Breaking has a long tradition in planning and many other sub-areas of computer science (Starke 1991; Emerson and Sistla 1996; Fox and Long 1999; Rintanen 2003; Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012). We use an extension to decoupled search, introduced by Gnad et al. (2017), which is build on *orbit search* (Domshlak, Katz, and Shleyfman 2015; Wehrle et al. 2015). An orbit is a set of states all of which

are symmetric to each other. In the search, each state is mapped to a canonical representative of its orbit. In case another state from the same orbit has already been generated, a new state can safely be pruned. *Decoupled orbit search* extends this concept to decoupled states.

Decoupled Dominance Pruning

Another extension that has recently been introduced is dominance pruning (Torralba et al. 2016), where decoupled states that are dominated by other – already generated – states can be safely discarded. We only deploy a very lightweight pruning method, namely *frontier* pruning. The standard way of performing duplicate checking in decoupled search 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 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 irrelevant leaf states and leaf actions. In some domains, this can tremendously reduce the size of the leaf state spaces.

The techniques described in this sub-section are only applicable if $\mathcal{F}$ is a fork factoring.

Implementation

Decoupled Search has been implemented as an extension of Fast Downward (FD) (Helmert 2006). The implementation does not support conditional effects. 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 SaarPlan are greedy best-first search (GBFS) and the h^{FF} heuristic (Hoffmann and Nebel 2001). Search algorithms and heuristics can be adapted to decoupled search using a compilation defined by Gnad and Hoffmann (2018). We will use the following notation to describe our techniques: the decoupled variant of search algorithm X is denoted $\mathbf{DX}$. We denote fork (inverted-fork) factorings by **F** (**IF**), and factorings generated using the incident-arcs algorithm by **IA**. To combine the power of the factoring strategies, we use a portfolio approach that runs multiple strategies and picks the one with the maximum number of leaf factors. Further more, we restrict the size for the per-leaf domain-size product to ensure that the leaf state spaces are reasonably small and do not incur a prohibitive runtime overhead when generating new decoupled states. We denote this size limit by $|F^L_{max}| := \max_{F^L \in \mathcal{F}^L} \Pi_{v \in F^L} |\mathcal{D}(v)|$, where $\mathcal{D}(v)$ denotes the domain of variable v. If a fork factoring is detected, we sometimes perform frontier dominance pruning, denoted **FP** and reduce the size of the leaf state spaces removing irrelevant transitions and states (**IP**). (Decoupled) orbit search is abbreviated **(D)OSS**.

Grey Planning

In the spirit of partial delete-relaxation methods, like red-black planning, which has been used in the Mercury planner (Katz and Hoffmann 2014), SaarPlan employs an extension thereof, *grey planning* (Speicher et al. 2017). In this paper, we only give a brief summary of grey planning and refer the reader to Speicher et al. (2017) for full details.

Partial delete-relaxation methods interpolate between delete-relaxed planning and real planning (Keyder, Hoffmann, and Haslum 2012; 2014; Katz, Hoffmann, and Domshlak 2013; Domshlak, Hoffmann, and Katz 2015). Red-black planning applies the delete-relaxed semantics to a subset of state variables (the "red" ones), letting them accumulate their values, while keeping the real semantics for the others (the "black" ones). It is tractable if the dependencies between black variables are acyclic, and each black variable is *invertible*. The heuristic function based on that tractable fragment, h^{RB}, was a key part of Mercury.

Distinctions at the level of entire state variables, however, are very coarse-grained: either we remember all past values of a variable (red), or only the most recent one (black). Grey planning uses *limited-memory* state variables, instead, that allow more fine-grained relaxations through remembering a subset of their most recent values. So it partially relaxes *within* state variables, remembering only a subset of the most recent values, as needed for tractability. Limited memory can be used to substantially extend the abovementioned tractable fragment. In h^{RB}, non-invertible variables cannot be painted black, because they may not be able to go back to a previous value when required. In grey planning, the idea is to give these variables "just enough" memory to ensure this property, instead of fully delete-relaxing them. The resulting heuristic function, h^{Gray}, has proven to improve over h^{RB} in many domains.

Implementation

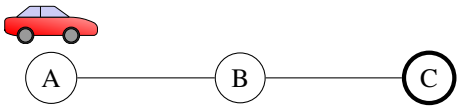
As our other methods, h^{Gray} is implemented in Fast Downward (Helmert 2006), adopting Domshlak et al.'s *stop search* technique, which tests whether the relaxed plan is actually a real plan, and if so, stops the search. In fact, we never run an actual search with h^{Gray}, but *only* use the stop search mechanism. If it succeeds in the initial state, we are done. Else, we stop the run and proceed with the next component. We also adopted the painting strategy of Domshlak, Hoffmann, and Katz (2015), which has been used in the Mercury planner (Katz and Hoffmann 2014). The main advantage of h^{Gray} over the red-black heuristic of Mercury lies in the additional stop-search prowess, thus adding another increment to that same main advantage of h^{RB} over h^{FF}. The abbreviation of our grey-planning "no-search" approach is **GREY**. Conditional effects are not supported.

Partial Delete Relaxation with $h^{C\text{FF}}$

Like grey planning, the $h^{C\text{FF}}$ heuristic is an approach to partial delete relaxation. The partially relaxed plans must respect a given set of conjunctions C, which represent combinations of facts that must be achieved *simultaneously* (Hoffmann and Fickert 2015; Fickert, Hoffmann, and Steinmetz

2016). Whenever a conjunction is a subset of the preconditions of an action, the conjunction of these facts must be achieved instead of the facts individually.

Consider the task illustrated below. The car has to move from A to C. The car can only hold one unit of fuel, which each drive action consumes, but can be refueled at any location. Formally, there are STRIPS facts *at(x)* for the position of the car and *fuel* to indicate if the car has fuel. Initially the car is at location A and holds fuel.



A fully delete relaxed plan can ignore the fuel consumption and just apply the drive actions from A to B and B to C immediately after each other. The critical conjunction of facts that is ignored here is that the car must be at B while holding fuel before the second drive action can be executed. This conjunction can not be achieved by any of the drive actions as they consume the fuel fact. A partially relaxed plan generated by the $h^{C\text{FF}}$ heuristic respecting this conjunction would have to add the refuel action before driving from B to C, making the relaxed plan a real plan. In fact, with a sufficiently large set of conjunctions C, all plans generated by $h^{C\text{FF}}$ are real plans.

Refinement-HC with Novelty Pruning

The $h^{C\text{FF}}$ heuristic works best when the conjunctions are generated online, in particular in Refinement-HC (**RHC**) (Fickert and Hoffmann 2017a), which is an extension of enforced hill-climbing (EHC) (Hoffmann and Nebel 2001). Like standard EHC, the algorithm progresses through iterations of breadth-first search (BrFS) until a state s with lower heuristic value is found, then search continues from there. In RHC, these explorations are bounded by a fixed depth. If a state s with lower heuristic value can not be found within that bound, the heuristic is refined and the BrFS phase is restarted. Thus, RHC escapes local minima through heuristic refinement instead of brute-force search. A second extension to standard EHC are restarts from the initial state (without resetting the heuristic) whenever the search is stuck in a dead end. Due to the convergence of the partially relaxed plans generated by $h^{C\text{FF}}$ to real plans, RHC is complete.

In SaarPlan, we use an extension of Refinement-HC where the refinement criterion is based on novelty pruning instead of a simple depth bound (Fickert 2018). Instead of using BrFS with bounded depth in the local exploration phase, we perform exhaustive BrFS with incomplete novelty pruning, similar to a single iteration IW(k) of iterated width search (Lipovetzky and Geffner 2012). In our setting, a state passes the novelty test if it contains at least one novel *conjunction* $c \in C$, otherwise it is pruned. This corresponds to IW(1), but uses the conjunctions of $h^{C\text{FF}}$ instead of the individual facts. The novelty pruning is only applied in the BrFS phase, and thus only considers the states in the current BrFS exploration for pruning, not across the overall search.

GBFS and Weighted A*

Refinement-HC can not always overcome the limitations of local search. For example in Sokoban, the presence of deep dead ends has proven difficult, and global search algorithms like GBFS are much more suitable here.

Given these drawbacks, we place a time limit on Refinement-HC, as well as a growth bound on the number of conjunctions for the $h^{C\text{FF}}$ heuristic and run GBFS after Refinement-HC terminates. The growth bound on $h^{C\text{FF}}$ is motivated by the observation that in domains where Refinement-HC works well, the set of conjunctions typically does not grow excessively large.

The GBFS phase uses a dual-queue of $h^{C\text{FF}}$ and a landmarks-count heuristic (Porteous, Sebastia, and Hoffmann 2001; Richter, Helmert, and Westphal 2008). This makes it is very similar to LAMA (Richter and Westphal 2010), using $h^{C\text{FF}}$ instead of h^{FF}. The set of conjunctions for the $h^{C\text{FF}}$ heuristic is reset to a fixed size bound before starting the GBFS phase. During search, the heuristic periodically replaces old conjunctions by newly generated ones, which allows it to adapt itself to the part of the search space that is currently being explored (Fickert and Hoffmann 2017b). Again, similar to LAMA, when GBFS finds a solution, search continues with an anytime phase of weighted A* with incrementally lower weights. We cache heuristic values across the GBFS and weighted A* iterations to reduce overhead.

Implementation

Unsurprisingly, $h^{C\text{FF}}$ and the related techniques are also implemented on top of FD (Helmert 2006). Similar to the stop search technique used in our grey planning component, here we stop search whenever no conflict could be found in the refinement process, which implies that the partially relaxed plan is also a real plan.

SaarPlan Configurations

SaarPlan combines the described techniques into a sequential portfolio. In addition to the standard FD preprocessor, we perform a relevance analysis based on h^2 to simplify the planning task prior to the search (Alcázar and Torralba 2015). The mutexes found in this process are also used by the $h^{C\text{FF}}$ heuristic to reduce its computational overhead.

This section describes configuration details for the individual tracks. We use the following abbreviations:

- **PO**: dual-queue for preferred operators.
- **HA**: helpful actions pruning.
- **N**: novelty pruning.

In all tracks, we start by ignoring the action costs. Costs are ignored altogether in the agile track, and only reintroduced in the bounded-cost track if no plan below the cost bound could be found. In the satisficing track, we reintroduce the real costs upon finding the first plan.

In the following sub-sections, we detail the configurations employed in each competition track. We provide the search configurations, as well as the time each of the components is allotted (in seconds).

Satisficing Track

The portfolio configuration for the satisficing track is shown in Figure 1. By default, all configurations ignore action costs, but reintroduce them after finding the first plan.

Search	Factoring	$\lvert F^L_{max}\rvert$	Heuristic	Pruning	Runtime
DGBFS	F	$1M$	h^{FF}	FP,IP,PO	$100s$
(D)GBFS	F/IF/IA	$1/1/0.1M$	h^{FF}	(D)OSS,PO	$600s$
GREY	-	-	h^{Gray}	-	$1100s$*
RHC	-	-	$h^{C\text{FF}}$	HA,N	$1100s$*
GBFS	-	-	h^{LM},$h^{C\text{FF}}$	PO	$1100s$*
WA*	-	-	h^{LM},$h^{C\text{FF}}$	PO	$1100s$*

Figure 1: Portfolio configuration in the satisficing track. Components are launched top to bottom. Components whose timeout is marked with * share their timeout. The RHC component also has an individual timeout of $500s$.

Saarplan starts with two decoupled search configurations. The first one runs decoupled search with a fork factoring, since these typically perform better, in particular when combined with the strong leaf pruning methods (FP,IP). The second component tries all factoring strategies, and additionally enables decoupled orbit search (DOSS). If none of the factoring strategies succeeds, the component falls back to standard search using the same options (indicated by the "D" in parantheses). Both components use the h^{FF} heuristic with preferred operators in a dual-queue.

After the decoupled search components, we first attempt to find a grey plan for the initial state and check if it is a real plan. Then, we run the components of the OLCFF planner (Fickert and Hoffmann 2018), starting with Refinement-HC with novelty pruning. The final phase is a LAMA-like anytime search with GBFS and weighted A* using incrementally lower weights, where the main difference to LAMA is the use of $h^{C\text{FF}}$ instead of h^{FF}.

Agile Track

The portfolio configuration for the agile track is shown in Figure 2. All configurations ignore action costs.

Search	Factoring	$\lvert F^L_{max}\rvert$	Heuristic	Pruning	Runtime
DGBFS	F	$1M$	h^{FF}	FP,IP,PO	$30s$
GREY	-	-	h^{Gray}	-	$170s$*
RHC	-	-	$h^{C\text{FF}}$	HA,N	$170s$*
GBFS	-	-	h^{LM},$h^{C\text{FF}}$	PO	$170s$*
(D)GBFS	F/IF/IA	$1/1/0.1M$	h^{FF}	(D)OSS,PO	$100s$

Figure 2: Portfolio configuration in the agile track. Components are launched top to bottom. Components whose timeout is marked with * share their timeout. The RHC component also has an individual timeout of $100s$.

In the agile track, we use a similar configuration to the satisficing track with only small differences. The second decoupled search configuration is moved to the end, and we don't need the weighted A* phase. Since the time limit is much lower in the agile track, the time limits of the individual components are reduced accordingly.

Bounded-Cost Track

The portfolio configuration for the bounded-cost track is shown in Figure 2. All components use normal action costs, except the first two which use unit action costs and only check if the cost-bound is satisfied upon finding a solution.

Search	Factoring	$\lvert F^L_{max}\rvert$	Heuristic	Pruning	Runtime
DGBFS	F	$1M$	h^{FF}	FP,IP,PO	$100s$
(D)GBFS	F/IF/IA	$1/1/0.1M$	h^{FF}	(D)OSS,PO	$600s$
GREY	-	-	h^{Gray}	-	$800s$*
RHC	-	-	$h^{C\text{FF}}$	HA,N	$800s$*
GBFS	-	-	h^{LM},$h^{C\text{FF}}$	PO	$800s$*
(D)WA*	F/IF/IA	$10/10/1M$	h^{LM},h^{FF}	(D)OSS,PO	$300s$

Figure 3: Portfolio configuration in the cost-bounded track. Components are launched top to bottom. Components whose timeout is marked with * share their timeout. The RHC component also has an individual timeout of $400s$.

Again, we use a similar configuration to the satisficing track. We replaced the anytime phase with a (decoupled) weighted A* with $w = 3$, using h^{LM} and h^{FF} and a dual-queue for preferred operators.

Conclusion

SaarPlan combines a set of powerful planning techniques into a sequential portfolio. This portfolio is designed in a way that quick-to-terminate methods, like star-topology decoupled search or grey planning's stop-search, are applied first, to find a plan as fast as possible. More search-heavy algorithms like the online-refining $h^{C\text{FF}}$ heuristic are executed later, in case the other methods fail. We augment our techniques with recently introduced extensions like novelty pruning, and symmetry breaking in decoupled search, to further spread the range of techniques.

Acknowledgments. This work was partially supported by the German Research Foundation (DFG), under grants HO 2169/5-1 ("Critically Constrained Planning via Partial Delete Relaxation") and 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 et al. (2015), 2–6.

Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds. 2012. *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press.

Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds. 2015. *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In Bonet et al. (2012).

Domshlak, C.; Katz, M.; and Shleyfman, A. 2015. Symmetry breaking in deterministic planning as forward search: Orbit space search algorithm. *Technical Report IS/IE-2015-02*.

Emerson, E. A., and Sistla, A. P. 1996. Symmetry and model-checking. *Formal Methods in System Design* 9(1/2):105–131.

Fickert, M., and Hoffmann, J. 2017a. Complete local search: Boosting hill-climbing through online heuristic-function refinement. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)* (2017).

Fickert, M., and Hoffmann, J. 2017b. Ranking conjunctions for partial delete relaxation heuristics in planning. In Fukunaga, A., and Kishimoto, A., eds., *Proceedings of the 10th Annual Symposium on Combinatorial Search (SOCS'17)*. AAAI Press.

Fickert, M., and Hoffmann, J. 2018. OLCFF: Online-learning $h^{C\text{FF}}$. In *IPC 2018 planner abstracts*.

Fickert, M.; Hoffmann, J.; and Steinmetz, M. 2016. Combining the delete relaxation with critical-path heuristics: A direct characterization. *Journal of Artificial Intelligence Research* 56(1):269–327.

Fickert, M. 2018. Making hill-climbing great again through online relaxation refinement and novelty pruning. In Bulitko, V., and Storandt, S., eds., *Proceedings of the 11th Annual Symposium on Combinatorial Search (SOCS'18)*. AAAI Press.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In Pollack, M., ed., *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, 956–961. Stockholm, Sweden: Morgan Kaufmann.

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

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry breaking in star-topology decoupled search. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)* (2017).

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.

Gnad, D.; Shleyfman, A.; and Hoffmann, J. 2018. DecStar - star-topology decoupled search at its best. In *IPC 2018 planner abstracts*.

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

Hoffmann, J., and Fickert, M. 2015. Explicit conjunctions w/o compilation: Computing $h^{\text{FF}}(\Pi^C)$ in polynomial time. In Brafman et al. (2015).

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

2017. *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, AAAI Press.

Katz, M., and Hoffmann, J. 2014. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC 2014 planner abstracts*, 43–47.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013. Who said we need to relax *all* variables? In Borrajo, D.; Fratini, S.; Kambhampati, S.; and Oddi, A., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, 126–134. Rome, Italy: AAAI Press.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In Bonet et al. (2012), 128–136.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2014. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research* 50:487–533.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In Raedt, L. D., ed., *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*, 540–545. Montpellier, France: IOS Press.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In Burgard, W., and Roth, D., eds., *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11)*. San Francisco, CA, USA: AAAI Press.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In Cesta, A., and Borrajo, D., eds., *Proceedings of the 6th European Conference on Planning (ECP'01)*, 37–48. Springer-Verlag.

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

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In Fox, D., and Gomes, C., eds., *Proceedings of the 23rd National Conference of the American Association for Artificial Intelligence (AAAI'08)*, 975–982. Chicago, Illinois, USA: AAAI Press.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In Giunchiglia, E.; Muscettola, N.; and Nau, D., eds., *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, 32–41. Trento, Italy: Morgan Kaufmann.

Speicher, P.; Steinmetz, M.; Gnad, D.; Hoffmann, J.; and Gerevini, A. 2017. Beyond red-black planning: Limited-memory state variables. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)* (2017), 269–273.

Starke, P. 1991. Reachability analysis of petri nets using symmetries. *Journal of Mathematical Modelling and Simulation in Systems Analysis* 8(4/5):293–304.

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.

Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Integrating partial order reduction and symmetry elimination for cost-optimal classical planning. In Yang, Q., ed., *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press/IJCAI.