DecStar – STAR-topology DECoupled Search at its best

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

Alexander Shleyfman
Technion - Israel Institute of Technology
Industrial Engineering & Management
Haifa, Israel
alesh@campus.technion.ac.il

Jörg Hoffmann
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
hoffmann@cs.uni-saarland.de

Abstract

DecStar extends Fast Downward by Star-Topology Decoupling (STD), a technique recently introduced in classical planning. It exploits independence between components of a planning task to reduce the size of the state-space representation. Partitioning the state variables into components, such that the interaction between these takes the form of a star topology, decoupled search only searches over action sequences affecting the center component of the topology, and enumerates reachable assignments to each leaf component, separately. This can lead to an exponential reduction in the search-space representation size. It is not always easy to find a partitioning for a given planning task, though, so we extend STD by a fallback option, that runs standard search whenever no (good) partitioning could be found.

Introduction

Star-Topology Decoupling (STD) is a recently introduced method to reduce the representation size of search spaces (Gnad and Hoffmann 2015; Gnad, Hoffmann, and Domshlak 2015; Gnad and Hoffmann 2018). 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 restricting the dependencies between the components, decoupled search has proven to be very efficient and competitive to state-of-the-art planners.

The performance of STD is highly influenced by the outcome of the *factoring* process, i. e., the process of finding a partitioning of the state variables. Just, how to find a good factoring, and what qualifies a factoring as being good? These questions have partially been answered by Gnad, Poser, and Hoffmann (2017), who devised two algorithms that can detect star topologies on a wide range of planning domains. Still, the proposed algorithms can *fail* to find a factoring, or succeed, but return a factoring with undesired properties, e. g. large leaf components that incur a prohibitive runtime overhead when generating new search states. In this case, we simply run standard search, instead [1].

When running STD, we enable some of the extensions that have been developed, namely partial-order reduction (POR) (Gnad, Wehrle, and Hoffmann 2016), symmetry breaking (Gnad et al. 2017), 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). Symmetry breaking has recently been introduced for decoupled search, too. It is a widely known approach across many areas of computer science (e. g. (Starke 1991; Emerson and Sistla 1996; Fox and Long 1999; Rintanen 2003; Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2012)). Dominance pruning identifies states that can be safely discarded, without affecting completeness (and optimality). POR and symmetry breaking can be used in any given factoring type [2], dominance pruning, however, is only applicable if the generated factoring takes the form of a fork, i. e., the leaves have dependencies on the center, but not vice versa.

In the fallback case, i. e., when no good factoring could be detected and we run standard search, we make use of the variety of techniques that are implemented in Fast Downward (Helmert 2006). In the optimal and bounded-cost tracks, this includes a pattern database heuristic generated using a genetic algorithm (Edelkamp 2006), the LM-cut heuristic (Helmert and Domshlak 2009), a Merge-&-Shrink heuristic (Helmert, Haslum, and Hoffmann 2007; Helmert et al. 2014), and a landmark-count heuristic (Porteous, Sebastia, and Hoffmann 2001; Richter, Helmert, and Westphal 2008). In the agile and satisficing tracks, we mostly use the h^{FF} heuristic (Hoffmann and Nebel 2001) and a search configuration similar to the LAMA planning system (Richter, Westphal, and Helmert 2011).

In all tracks, we extend the standard preprocessor of Fast Downward by the h^2-based task simplification of Alcázar and Torralba (2015), which removes irrelevant and unreachable facts and actions from the task.

[1] This limitation is merely due to the factoring strategies that we use to identify suitable partitionings. In general, *every* task has a star topology and can be tackled by decoupled search.

[2] We use a so-far unpublished extension of strong stubborn sets that supports general star factorings

Preliminaries

We use a finite-domain state variable formalization (FDR) of planning (e. g. (Bäckström and Nebel 1995; Helmert 2006)), where a planning 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 $\mathsf{cost}(a)$ is a's non-negative *cost*.

We use the usual FDR semantics. The *planning problem* is to decide if there exists a sequence of actions that transforms the *initial state* I of Π to a state that satisfies the *goal* condition G. In the optimal and bounded-cost tracks of the competition, we are looking for an action sequence with minimal, respectively bounded, summed-up cost.

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 $\mathcal{V}(\mathsf{pre}(a)) \subseteq F \cup F^C$. 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 (with lower g-cost), the new state can safely be pruned. *Decoupled orbit search* extends this concept to decoupled states.

Decoupled Strong Stubborn Sets

Partial-order reduction is a well-known technique that reduces the size of the search space by pruning transitions that correspond to different permutations of actions (Valmari 1989; Godefroid and Wolper 1991; Edelkamp, Leue, and Lluch-Lafuente 2004; Alkhazraji et al. 2012; Wehrle et al. 2013; Wehrle and Helmert 2014). A variant of strong stubborn sets, decoupled strong stubborn sets (DSSS), has also been introduced for decoupled search. We will employ DSSS in the optimal and bounded-cost tracks. For fork factorings, we use DSSS as defined by Gnad, Wehrle, and Hoffmann (2016). For non-fork factorings, we use a yet unpublished extension that is able to handle arbitrary factorings. To avoid the runtime overhead when DSSS are not effective, we implemented a "safety belt" mechanism, that disables DSSS if after the first 1000 expansions less than 20% of the transitions have been pruned.

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 (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 irrelevant leaf states and leaf actions. In some domains, this can tremendously reduce the size of the leaf state spaces.

As indicated before, the techniques described in this subsection are only applicable if $\mathcal{F}$ is a fork factoring.

Implementation & Configurations

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 DecStar planner are the A* search algorithm, and the $h^{\text{LM-cut}}$ heuristic (Helmert and Domshlak 2009) for optimal, and bounded-cost planning. In the satisficing and agile tracks, we run greedy best-first search (GBFS) using the h^{FF} heuristic (Hoffmann and Nebel 2001). The search algorithms and heuristics can be adapted to decoupled search using a compilation defined by Gnad and Hoffmann (2018). Our implementation does not support conditional effects. On top of the standard FD preprocessor, we perform a relevance analysis based on h^2, to eliminate actions and simplify the planning task prior to the search (Alcázar and Torralba 2015).

In all tracks of the competition, star-topology decoupling is the main component of our planner. However, since, as outline before, our factoring strategies are not guaranteed to find good task decompositions, we need a fallback method. Given the implementation of decoupled search in FD, we can easily make use of the many techniques that FD ships with. Thus, in the case that no good factoring could be obtained, we run standard search using some heuristics and pruning methods that are implemented in FD.

We will use the following notation to describe our techniques: the decoupled variant of search algorithm X is denoted **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 $|F^L_{max}| := \max_{F^L \in \mathcal{F}^L} \Pi_{v \in F^L} |\mathcal{D}(v)|$. 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 strong stubborn sets will be abbreviated as **DSSS**, where we always use the safety belt with a minimum pruning ratio of 20%. In standard search, the use of strong stubborn sets pruning is denoted **SSS**. (Decoupled) orbit search is abbreviated **(D)OSS**. The use of preferred operator pruning is denoted **PO**.

In all but the optimal track, we start by ignoring the action costs. Costs are ignored altogether in the agile track, and only re-introduced in the bounded-cost track if no plan below the cost bound could be found. In the satisficing track, we re-introduce 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).

Optimal Track

DecStar starts by running decoupled search with a fork factoring with a maximum leaf size of 10 million, if one exists. In this case, it employs frontier pruning, removes irrelevance in the leaves, and performs partial-order reduction (DSSS). The next component tries all factoring methods with different size constraints, and prunes states with DSSS and DOSS. This is the main component running for 15min. Both decoupled search components use the LM-cut heuristic (Helmert and Domshlak 2009), currently the strongest admissible heuristic that supports decoupled search.

Search	Factoring	$\|F^L_{max}\|$	Heuristic	Pruning	Runtime
DA^*	F	$10M$	$h^{\text{LM-cut}}$	DSSS,FP,IP	$100s$
DA^*	F/IF/IA	$10/10/1M$	$h^{\text{LM-cut}}$	DSSS,DOSS	$800s$
A*	-	-	$h^{\text{LM-cut}}$	SSS,OSS	$180s$
A*	-	-	$h^{\text{GA-PDB}}$	SSS	$180s$
A*	-	-	$h^{\text{M\&S}}$	-	$180s$
A*	-	-	h^{LMc}	-	$180s$
A*	-	-	blind	-	$180s$

Figure 1: Portfolio configuration in the optimal track. Components are launched top to bottom.

In case no matching factoring could be found, or when decoupled search fails, DecStar is supported by standard search with different heuristics. If the heuristic does not support conditional effects, we also enable strong stubborn sets pruning and/or orbit search, which both do not support these, either. DecStar tries the pattern database heuristic with patterns generated using a genetic algorithm ($h^{\text{GA-PDB}}$) (Edelkamp 2006), a Merge&Shrink heuristic with linear merge order and bisimulation ($h^{\text{M\&S}}$) (Helmert, Haslum, and Hoffmann 2007; Helmert et al. 2014; Sievers, Wehrle, and Helmert 2014), the landmark-count heuristic (h^{LMc}) (Porteous, Sebastia, and Hoffmann 2001; Richter, Helmert, and Westphal 2008), and finally blind search.

Satisficing Track

In the satisficing track, DecStar runs three different components. The first, similar to the optimal track, 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. The "D" in paranthesis indicates that, if none of the factoring strategies succeeds, the component falls back to standard search using the same options. Both components use the h^{FF} heuristic and perform preferred operator pruning, using FD's dual queue mechanism.

Search	Factoring	$\|F^L_{max}\|$	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	$1000s$
$GBFS$	-	-	h^{LM},h^{FF}	PO	$700s$

Figure 2: Portfolio configuration in the satisficing track. Components are launched top to bottom.

If all is lost, DecStar gets help from his experienced friend LAMA, adopting its first iteration (Richter, Westphal, and Helmert 2011).

Bounded-Cost Track

The components that DecStar uses in the bounded-cost track are a mix of the components described above for the optimal and satisficing track. DecStar starts by running each satisficing-track component for 100s. It then uses a weighted A^* search (weight 3), in case none of the previous components could find a plan within the given bound. This is followed by the components used in the optimal track.

Search	Factoring	$\lvert F^L_{max}\rvert$	Heuristic	Pruning	Runtime
$DGBFS$	F	$1M$	h^{FF}	PO,FP,IP	$100s$
$(D)GBFS$	F/IF/IA	$1/1/0.1M$	h^{FF}	(D)OSS,PO	$100s$
$GBFS$	-	-	h^{LM},h^{FF}	PO	$100s$
$DW\text{A}^*$	F/IF/IA	$10/10/1M$	h^{FF}	DOSS	$400s$
$D\text{A}^*$	F	$10M$	$h^{\text{LM-cut}}$	DSSS,FP,IP	$100s$
$D\text{A}^*$	F/IF/IA	$10/10/1M$	$h^{\text{LM-cut}}$	DSSS,DOSS	$400s$
A^*	-	-	$h^{\text{LM-cut}}$	SSS,OSS	$120s$
A^*	-	-	$h^{\text{GA-PDB}}$	SSS	$120s$
A^*	-	-	$h^{\text{M\&S}}$	-	$120s$
A^*	-	-	h^{LMc}	-	$120s$
A^*	-	-	blind	-	$120s$

Figure 3: Portfolio configuration in the bounded-cost track. Components are launched top to bottom.

Agile Track

Search	Factoring	$\lvert F^L_{max}\rvert$	Heuristic	Pruning	Runtime
$DGBFS$	F	$10K$	h^{FF}	FP,IP,PO	$60s$
$(D)GBFS$	F/IF/IA	$10/10/1K$	h^{FF}	(D)OSS,PO	$120s$
$GBFS$	-	-	h^{LM},h^{FF}	PO	$120s$

Figure 4: Portfolio configuration in the agile track. Components are launched top to bottom.

In the agile track, DecStar uses the same search components as in the satisficing track, but with different timeouts and leaf space size limits. The latter is due to the fact that the larger the leaves, the bigger the runtime overhead per decoupled state. Since the time limit is significantly smaller in the agile track, we try to keep the leaves as small as possible. In spite of the tight time constraint, we still run the h^2-preprocessor for 10s.

Conclusion

DecStar is the best that star-topology decoupling currently has to offer. Many extensions have been developed, allowing the use of various search algorithms, heuristic functions, and pruning techniques. Decoupled search has proved to be a method that can beat other state-of-the-art planners, also in the unsolvability IPC 2014, if the given planning task can be nicely decoupled. Even outside the planning area, namely in proving safety properties in model checking, star-topology decoupling has shown its merit (Gnad et al. 2018).

And still, there are many possible ways of further extending it. In classical planning, where it is crucial to use strong heuristics, the next steps are to do research on how to apply abstraction and LP heuristics in decoupled search. In model checking, an interesting research question is the extension of star-topology decoupling to prove liveness properties.

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)*, 2–6. 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.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. 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.

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*.

Edelkamp, S.; Leue, S.; and Lluch-Lafuente, A. 2004. Partial-order reduction and trail improvement in directed model checking. *International Journal on Software Tools for Technology Transfer* 6(4):277–301.

Edelkamp, S. 2006. Automated creation of pattern database search heuristics. In *Proceedings of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt 2006)*, 35–50.

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

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. 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.; 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)*. AAAI Press.

Gnad, D.; Dubbert, P.; Lluch-Lafuente, A.; and Hoffmann, J. 2018. Star-topology decoupling in spin. In del Mar Gallardo, M., and Merino, P., eds., *Proceedings of the 25th International Symposium on Model Checking of Software (SPIN'18)*, Lecture Notes in Computer Science. Springer.

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.

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)*, 3110–3116. AAAI Press/IJCAI.

Godefroid, P., and Wolper, P. 1991. Using partial orders for the efficient verification of deadlock freedom and safety properties. In *Proceedings of the 3rd International Workshop on Computer Aided Verification (CAV'91)*, 332–342.

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.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery* 61(3).

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Boddy, M.; Fox, M.; and Thiebaux, S., eds., *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, 176–183. Providence, Rhode Island, USA: Morgan Kaufmann.

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.

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.; 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.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011 (planner abstract). In *IPC 2011 planner abstracts*, 50–54.

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.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In Brodley, C. E., and Stone, P., eds., *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 2358–2366. Austin, Texas, USA: AAAI Press.

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.

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.

Wehrle, M.; Helmert, M.; Alkhazraji, Y.; and Mattmüller, R. 2013. The relative pruning power of strong stubborn sets and expansion core. In Borrajo, D.; Fratini, S.; Kambhampati, S.; and Oddi, A., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*. Rome, Italy: AAAI Press.

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.