DiSCO: Decoupled Search + COnjunctions

Maximilian Fickert[1], **Daniel Gnad**[2]

[1] Independent Researcher
[2] Linköping University, Sweden
fickert@cs.uni-saarland.de, daniel.gnad@liu.se

Abstract

DiSCO combines two main components in a portfolio approach: Decoupled state-space Search and COnjunctions heuristics. Decoupled search exploits the structure of the problem to decompose planning tasks. This allows a compact state-space representation that makes the search more efficient. Conjunctions heuristics are a generalization of delete relaxation, that "unrelax" sets of fact conjunctions that can be refined during search. Both techniques individually have shown to yield state-of-the-art performance in classical planning. In DiSCO, we combine them in a sequential portfolio to inherit their respective strengths.

Introduction

With *decoupled search* and *conjunctions heuristics*, DiSCO combines two state-of-the-art techniques into a portfolio planner that participates in the satisficing and agile tracks of the competition.

Decoupled search tries 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.

In addition to decoupled search, DiSCO uses online-refinement search algorithms that improve the semi-delete relaxation heuristic $h^{C\text{FF}}$ during search. It uses search algorithms based on enforced hill-climbing (EHC) and greedy best-first search (GBFS) with novelty pruning and relaxed subgoal counting. A LAMA-like anytime search, replacing the fully delete-relaxed h^{FF} heuristic with $h^{C\text{FF}}$, completes DiSCO on the satisficing track.

Finally, on tasks with ADL features such as conditional effects or axioms, DiSCO runs a portfolio of alternative configurations that drop advanced techniques in favor of simpler but feature-complete search algorithms and heuristics.

Decoupled Search

We perform decoupled search as 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* factorings from Gnad, Poser, and Hoffmann (2017), as well as the linear-programming (LP) based bM80s factoring method from Gnad, Torralba, and Fišer (2022). The outcome of the factoring process is a partitioning $\mathcal{F} = \langle C, \mathcal{L} \rangle$ of the variables of the planning task Π, where C is the (possibly empty) *center factor* of $\mathcal{F}$ and $\mathcal{L}$ is its set of non-empty *leaf factors*, such that $C \cup \bigcup_{L \in \mathcal{L}} L = V$. A factoring $\mathcal{F}$ induces a partitioning of the actions into *global actions* $\mathcal{A}^G$ and *leaf actions* $\mathcal{A}^{\mathcal{L}}$. For every $L \in \mathcal{L}$, the leaf actions $\mathcal{A}^L$ of L are those actions that have effects only on L and are preconditioned by variables in $C \cup L$. We define the set of leaf actions as $\mathcal{A}^{\mathcal{L}} := \bigcup_{L \in \mathcal{L}} \mathcal{A}^L$ and the global actions as $\mathcal{A}^G := \mathcal{A} \setminus \mathcal{A}^{\mathcal{L}}$.

Given a factoring $\mathcal{F}$, decoupled search is performed as follows: The search will only branch over global actions. Along such a path of global actions π^G, for each leaf factor L, the search maintains a set of leaf paths, i. e., actions only affecting variables of L, that *comply* with π^G. Intuitively, for a leaf path π^L to comply with a global path π^G, it must be possible to embed π^L into π^G into an overall action sequence π, such that π is a valid path in the projection of the planning task Π onto $C \cup L$. A decoupled state corresponds to an end state of such a global 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 global 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.

Because decouples states represent sets of explicit states, exact duplicate checking is less effective in decoupled search. We employ dominance pruning instead, which we describe next.

Decoupled Dominance Pruning

Dominance pruning (Torralba et al. 2016) can be generalized to decoupled search such that 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 aforementioned techniques are only applicable if $\mathcal{F}$ is a fork factoring. For all factorings, we employ the dominance pruning enhancements introduced in Gnad (2021).

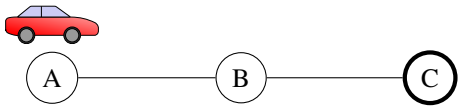
Implementation

Decoupled Search has been implemented as an extension of Fast Downward (FD) (Helmert 2006). The implementation does not support conditional effects, derived predicates, and axioms. 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 DiSCO 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 **DX**. We denote fork (inverted-fork) factorings by **F** (**IF**), and factorings generated using the LP-based algorithm by **LP**. We impose a time limit of 30 seconds for the factoring process. 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 $|L_{max}| := \max_{L \in \mathcal{L}} \Pi_{v \in L} |\mathcal{D}(v)|$, 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**).

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

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, each representing a combination 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.

Online-Refinement Search Algorithms

The $h^{C\text{FF}}$ heuristic works best when the conjunctions are generated online, ideally in a search algorithms specialized for online refinement. One such algorithm is Refinement-HC (RHC) (Fickert and Hoffmann 2017a, 2022), 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.

RHC has been extended to use incomplete novelty pruning instead of a simple depth bound in the local exploration phase (Fickert 2018; Fickert and Hoffmann 2022), similar to a single iteration IW(k) of iterated width search (Lipovetzky and Geffner 2012). Another extension, RHC-SC, uses relaxed subgoal counting (Lipovetzky and Geffner 2014) in the local exploration, replacing expensive relaxed-plan computations with cheap counting of achieved subgoals along the relaxed plan (Fickert and Hoffmann 2022). Finally, this idea can be transferred to greedy-best first search, resulting in the online-refinement search algorithm GBFS-SCL (Fickert 2020; Fickert and Hoffmann 2022).

Algorithms in DiSCO

In DiSCO, we use variants of both RHC-SC and GBFS-SCL. The algorithms use incomplete-novelty pruning on conjunction level, i.e., each local lookahead search prunes states that do not contain at least one novel conjunction among all states seen in this local exploration (not across the overall search).

In the agile-track configuration, our planner includes a novel variant $h^{\text{SC}}_{\text{add}}$ of the subgoal counting heuristic h^{SC} that is based on the h^{add}-values of the subgoals (extended to con-

junctions when used with $h^{C\text{FF}}$). More specifically: the standard subgoal-counting heuristic h^{SC} counts the number of achieved subgoals along the path from some root state s to current state s'; our h^{add}-variant sums up the $h^{\text{add}}(s, g)$-values of subgoals $g \subseteq s'$ instead (only considering the current state s', not any other state along the path from s).

In the satisficing track, our last configuration is variant of LAMA using a dual-queue of $h^{C\text{FF}}$ and a landmark-counting heuristic (Porteous, Sebastia, and Hoffmann 2001; Richter, Helmert, and Westphal 2008). The set of conjunctions for the $h^{C\text{FF}}$ heuristic is initially generated to a fixed size, and during search old conjunctions are periodically replaced by new ones to adapt to newly encountered states (Fickert and Hoffmann 2017b). The anytime search starts with GBFS and continues with incrementally decreasing weights in weighted A*, caching heuristic values across these searches to reduce computational overhead.

All algorithms are implemented on top of FD (Helmert 2006). The $h^{C\text{FF}}$-based algorithms are configured to stop search whenever the partially relaxed plan is a real plan.

DiSCO on ADL Tasks

Neither the $h^{C\text{FF}}$ heuristic nor Decoupled Search support advanced features such as conditional effects or axioms. Instead of solely relying on the provided automatic translation to STRIPS, we implemented variants of this portfolio that are used if the Fast Downward translator detects such features in the task.

Our ADL portfolios include a non-online-refinement variant of GBFS-SCL and, an old IPC classic, YAHSP (Vidal 2004, 2011). Both of these use the standard h^{FF} heuristic as implemented in Fast Downward, which has the necessary support for ADL features. We also include a LAMA-style configuration using a best-first search with an alternation queue of h^{FF} and a landmark-counting heuristic.

DiSCO Configurations

DiSCO combines the described techniques into a sequential portfolio. This section describes configuration details for the individual tracks. In addition to the standard FD translator, 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. We use it only in the satisficing track with a time limit of 3 min, and only if the translator does not produce axioms. All search algorithms are configured to use a dual-queue for preferred operators with the h^{FF} or $h^{C\text{FF}}$ heuristic, the RHC-SC configurations in the agile track use helpful actions pruning instead.

In the satisficing track we start by ignoring the action costs and re-introduce them upon finding the first plan. Costs are ignored altogether in the agile track.

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 configurations for the satisficing track are shown in Figures 1 (STRIPS) and 2 (ADL). Only the final configuration uses action costs, all others use unit action costs.

Timeout (s)	Search	Heuristics	Notes
50/50	DGBFS	h^{FF}	IF, $\|L_{max}\| = 100k$
50/0	DGBFS	h^{FF}	F, $\|L_{max}\| = 100k$, FP, IP
1000/180	GBFS-SCL	$h^{C\text{FF}}$	h^{SC}
600/0	DGBFS	h^{FF}	LP, $\|L_{max}\| = 2^{31} - 1$
*	GBFS/WA*	$h^{C\text{FF}}$/h^{LM}	

Figure 1: STRIPS portfolio configuration in the satisficing track. Components are launched top to bottom. In the timeout column, the first number indicates the timeout if no solution is found yet, the second number indicates the timeout if we seek to improve a previously found solution. The last configuration runs until the solution is proved optimal or the overall timeout is reached.

Timeout (s)	Search	Heuristics
720/720	GBFS-SCL	h^{FF}
480/0	YAHSP	h^{FF}
*	GBFS/WA*	h^{FF}/h^{LM}

Figure 2: ADL portfolio configuration in the satisficing track, see Figure 1 for explanations.

For STRIPS planning tasks, DiSCO starts with two decoupled search configurations. The first one runs decoupled search with an inverted-fork factoring, since these typically perform better. The second component uses fork factorings.

After the initial decoupled search components, we run GBFS-SCL with online refinement of the $h^{C\text{FF}}$ heuristic. If that does not succeed, we try another iteration of decoupled search using the LP-based factoring with the same search configuration as before. 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} and a cross-iteration cache of heuristic values.

For ADL tasks, DiSCO uses GBFS-SCL with h^{FF}, followed by YAHSP and LAMA.

Agile Track

The portfolio configurations for the agile track are shown in Figure 3 and 4. All configurations ignore action costs.

In the agile track, we use similar configurations to the ones from the satisficing track with only small differences. We add RHC-SC into the mix, and add another GBFS-SCL configuration using $h^{\text{SC}}_{\text{add}}$ as a potentially greedier variant to run in the beginning. For STRIPS tasks, we drop the LAMA-like configuration as we don't need to improve plans. Since the time limit is much lower in the agile track, the time limits of the individual components are reduced accordingly. In both variants, we try several techniques using a very short

Timeout (s)	Search	Heuristics	Notes
3	GBFS-SCL	$h^{C\text{FF}}$	$h^{\text{SC}}_{\text{add}}$
3	DGBFS	h^{FF}	IF, $\lvert L_{max}\rvert = 100k$
3	DGBFS	h^{FF}	F, $\lvert L_{max}\rvert = 100k$, FP, IP
3	RHC-SC	$h^{C\text{FF}}$	$h^{\text{SC}}_{\text{add}}$
90	GBFS-SCL	$h^{C\text{FF}}$	h^{SC}
120	DGBFS	h^{FF}	LP, $\lvert L_{max}\rvert = 2^{31} - 1$
*	RHC-SC	$h^{C\text{FF}}$	h^{SC}

Figure 3: STRIPS portfolio configuration in the agile track. Components are launched top to bottom. The last configuration runs until the overall timeout is reached.

Timeout (s)	Search	Heuristics	Notes
2	GBFS-SCL	h^{FF}	$h^{\text{SC}}_{\text{add}}$
2	YAHSP	h^{FF}	
2	GBFS	h^{FF}/h^{LM}	
120	GBFS-SCL	h^{FF}	h^{SC}
90	YAHSP	h^{FF}	
*	GBFS	h^{FF}/h^{LM}	

Figure 4: ADL portfolio configuration in the agile track, see Figure 3 for explanations.

timeout to see if they solve the task very quickly. If that fails, we give more time to a set of selected configurations.

Post-Competition Analysis

We analyzed the competition results by investigating the log files that have been provided by the organizers. Table 1 summarizes our findings in a per-domain analysis for all competition domains in both tracks, agile (left) and satisficing (right). The first column in each half, $\mathcal{F}$, shows the number of instances in which a factoring was detected, so decoupled search was in play. A first observation is that there are only three domains, quantum-layout, recharging-robots, and ricochet-robots, in which decoupled search can actually be performed. This is a quite low number compared to previous IPCs, so the results are dominated by explicit-state search.

The table shows detailed coverage results, distinguishing the components of each portfolio. We show, from left to right, the components in their order of execution for every portfolio. Hence, for example the first "SCL" column in the STRIPS portfolio of the agile track corresponds to GBFS-SCL with $h^{C\text{FF}}$ using $h^{\text{SC}}_{\text{add}}$ (cf. Figure 3). We indicate the decoupled-search configurations by the factoring they use, F, IF, LP, and use LA as an acronym for the LAMA-like configurations, adding a superscript "C" if $h^{C\text{FF}}$ is used instead of the default h^{FF}.

In the agile track, the important components are the variants of GBFS-SCL, contributing 53 instances in the STRIPS portfolio, and 13 in ADL. In ADL, the YAHSP component solved the highest number of instances (31). In the satisficing track, decoupled search with fork factoring (F), and again GBFS-SCL with $h^{C\text{FF}}$ performed strong, with a coverage of 12, respectively 65. In the ADL portfolio, GBFS-SCL and YAHSP contributed significantly to coverage, with 17 respectively 30 instances. We remark that, in rubiks-cube, since the "norm" version of the domain has conditional effects, we invoked our ADL portfolio for all instances.

Conclusion

DiSCO combines a set of powerful planning techniques into a sequential portfolio. This portfolio is designed in a way that quick-to-terminate methods, like decoupled 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 to further spread the range of techniques.

Acknowledgments

As we build on Fast Downward, we would like to thank all of its contributors. A big thank you also goes to the competition organizers, Daniel Fišer and Florian Pommerening.

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.

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)*, 158–162. AAAI Press.

Fickert, M. 2020. A Novel Lookahead Strategy for Delete Relaxation Heuristics in Greedy Best-First Search. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, 119–123. AAAI Press.

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)*, 107–115. AAAI Press.

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)*, 38–46. AAAI Press.

Fickert, M.; and Hoffmann, J. 2022. Online Relaxation Refinement for Satisficing Planning: On Partial Delete Relaxation, Complete Hill-Climbing, and Novelty Pruning. *Journal of Artificial Intelligence Research*, 73: 67–115.

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.

Gnad, D. 2021. Revisiting Dominance Pruning in Decoupled Search. In Leyton-Brown, K.; and Mausam, eds., *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*. AAAI Press.

			Agile Track															Satisficing Track									
			STRIPS Portfolio							ADL Portfolio								STRIPS Portfolio						ADL Portfolio			
Domain	#	$\mathcal{F}$	SCL	IF	F	RHC	SCL	LP	RHC	SCL	YA	LA	SCL	YA	LA	$\sum$	$\mathcal{F}$	IF	F	SCL	LP	LA^C	SCL	YA	LA	$\sum$
folding	20	0	1	0	0	0	6	0	2	0	0	0	0	0	0	9	0	0	0	10	0	0	0	0	0	10
folding^N	20	0	1	0	0	1	5	0	4	0	0	0	0	0	0	11	0	0	0	8	0	0	0	0	0	8
labyrinth	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
quantum-layout	20	1	18	0	1	1	0	0	0	0	0	0	0	0	0	20	15	0	12	7	1	0	0	0	0	20
recharge-robots	20	0	1	0	0	1	2	0	0	4	1	0	1	1	0	11	0	0	0	4	0	0	6	1	1	12
recharge-robots^N	20	1	3	0	0	0	10	0	1	0	0	0	0	0	0	14	1	0	0	14	0	0	0	0	0	14
ricochet-robots	20	17	0	0	0	0	3	0	0	0	0	0	0	0	0	3	0	0	0	19	0	0	0	0	0	19
rubiks-cube	20	0	0	0	0	0	0	0	0	4	1	0	0	15	0	20	0	0	0	0	0	0	6	14	0	20
rubiks-cube^N	20	0	0	0	0	0	0	0	0	4	1	0	0	15	0	20	0	0	0	0	0	0	5	15	0	20
slitherlink	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
slitherlink^N	20	0	0	0	0	1	3	0	0	0	0	0	0	0	0	4	0	0	0	3	0	2	0	0	0	5
$\sum$	220	19	24	0	1	4	29	0	7	12	3	0	1	31	0	112	16	0	12	65	1	2	17	30	1	128

Table 1: Per-domain and per-component analysis of the coverage in both tracks. $\mathcal{F}$ shows the number of instances on which a non-trivial factoring was detected. The $\sum$ columns show summed up per-domain coverage for the entire track across components. In all four portfolios, the components are listed from left to right in the execution order within the respective portfolio.

Gnad, D.; and Hoffmann, J. 2018. Star-Topology Decoupled State Space Search. *Artificial Intelligence*, 257: 24 – 60.

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)*, 4310–4316. AAAI Press/IJCAI.

Gnad, D.; Torralba, Á.; and Fišer, D. 2022. Beyond Stars – Generalized Topologies for Decoupled Search. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*. AAAI Press.

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

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

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.

Lipovetzky, N.; and Geffner, H. 2014. Width-based Algorithms for Classical Planning: New Results. In Schaub, T., ed., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, 1059–1060. Prague, Czech Republic: IOS 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.

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)*, 3265–3271. AAAI Press/IJCAI.

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.

Vidal, V. 2004. A Lookahead Strategy for Heuristic Search Planning. In Koenig, S.; Zilberstein, S.; and Koehler, J., eds., *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, 150–160. Whistler, Canada: AAAI Press.

Vidal, V. 2011. YAHSP2: Keep It Simple, Stupid. In *IPC 2011 planner abstracts*, 83–90.