Scorpion

Jendrik Seipp
University of Basel
Basel, Switzerland
jendrik.seipp@unibas.ch

This planner abstract describes “Scorpion”, the planner we submitted to the sequential optimization track of the International Planning Competition 2018. Scorpion is implemented in the Fast Downward planning system (Helmert 2006). It uses A* (Hart, Nilsson, and Raphael 1968) with an admissible heuristic (Pearl 1984) to find optimal plans. The overall heuristic is based on component abstraction heuristics that are combined by saturated cost partitioning (Seipp and Helmert 2018).[1]

In this abstract we only list the components of Scorpion and the settings we used for them. For a detailed description of the underlying algorithms we refer to Seipp (2018).

Abstraction Heuristics

Depending on whether or not a given task contains conditional effects, we use a different set of abstraction heuristics.

Tasks Without Conditional Effects

For tasks without conditional effects we use the combination of the following heuristics:

- Cartesian abstraction heuristics (CART):
  We consider Cartesian abstractions of the landmark and goal task decompositions (Seipp and Helmert 2018). We limit the total number of non-looping transitions in all abstractions underlying the Cartesian heuristics by one million.
- pattern databases found by hill climbing (HC):
  We use the algorithm by Haslum et al. (2007) for searching via hill climbing in the space of pattern collections. We limit the time for hill climbing by 100 seconds.
- pattern databases for systematic patterns (SYS):
  We use a procedure that generates all *interesting* patterns up to size 2 (Pommerening, Röger, and Helmert 2013).

Tasks With Conditional Effects

For tasks with conditional effects we compute pattern database heuristics for systematically generated patterns of sizes 1, 2 and 3 (Pommerening, Röger, and Helmert 2013). Since generating these heuristics can take very long for some tasks, we limit the time for generating PDB heuristics by 300 seconds.

Saturated Cost Partitioning

We combine the information contained in the component heuristics with saturated cost partitioning (Seipp and Helmert 2018). Given an ordered collection of heuristics, saturated cost partitioning iteratively assigns each heuristic h only the costs that h needs for justifying its estimates and saves the remaining costs for subsequent heuristics. Distributing the operator costs among the component heuristics in this way makes the sum of the individual heuristic values admissible.

The quality of the resulting saturated cost partitioning heuristic strongly depends on the order in which the component heuristics are considered (Seipp, Keller, and Helmert 2017). Additionally, we can obtain much stronger heuristics by maximizing over multiple saturated cost partitioning heuristics computed for different orders instead of using a single saturated cost partitioning heuristic (Seipp, Keller, and Helmert 2017). We therefore iteratively sample a state (using the sampling algorithm by Haslum et al. 2007), use a greedy algorithm for finding an initial order for the state (more concretely, we use the static greedy ordering algorithm with the $q_{\frac{h}{stolen}}$ scoring function) and afterwards optimize the order with simple hill climbing in the space of orders for at most two seconds (Seipp 2018). If the the saturated cost partitioning heuristic computed for the resulting optimized greedy order yields a higher estimate for one of a set of 1000 sample states than all previously added orders, we add the order to our set of orders. We limit the time for finding orders in this way to 200 seconds.

Operator Pruning Techniques

We employ two operator pruning techniques:

- strong stubborn sets:
  We use the variant that instantiates strong stubborn sets for classical planning in a straight-forward way (Alkhazraji et al. 2012; Wehrle and Helmert 2014). We compute the interference relation “on demand” during the search and switch off pruning completely in case the fraction of pruned successor states is less than 20% of the total successor states after 1000 expansions.

[1] We chose the name “Scorpion” since it contains the letters s(aturated) c(ost) p(artitioning) in this order.

Coverage	CART	HC	SYS	HC+CART	SYS+CART	SYS+HC	SYS+HC+CART
Agricola (20)	0	**4**	1	1	0	**4**	1
Data-Network (20)	**14**	12	12	**14**	**14**	12	**14**
Organic-Synthesis (20)	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Organic-Synthesis-Split (20)	**13**	**13**	12	**13**	**13**	**13**	**13**
Petri-Net-Alignment (20)	3	0	**7**	0	5	0	0
Snake (20)	11	**13**	**13**	**13**	**13**	**13**	**13**
Spider (20)	13	**15**	**15**	**15**	**15**	**15**	**15**
Termes (20)	12	13	12	**14**	12	**14**	**14**
Sum (160)	73	77	**79**	77	**79**	78	77

Table 1: Coverage scores of saturated cost partitioning over different heuristic subsets for IPC 2018 tasks that have no conditional effects after the translation phase.

- h^2 mutexes (Alcázar and Torralba 2015):
  This operator pruning method can remove irrelevant operators. We invoke it after translating a given input task to SAS$^+$ and before starting the search component of Fast Downward.

Post-IPC Evaluation

After the IPC 2018, we ran an experiment to analyze how much value each of the three sets of heuristics (CART, HC and SYS) contributes to the overall heuristic on the IPC 2018 benchmarks that have no conditional effects after the translation phase. We used a time and memory limit of 30 minutes and 7 GiB. Table 1 shows coverage results.

The seven different combinations of heuristics lead to similar total coverage scores (73–79 tasks). Using Cartesian heuristics (CART) leads to solving the lowest number of tasks, whereas using systematic PDBs by themselves (SYS) or combined with Cartesian abstractions (SYS+CART) achieves the maximal total coverage score. While coverage never decreases when adding systematic PDBs to the set of heuristics, it varies between domains whether the other types of heuristics are beneficial.

For the tasks in the Agricola domain, hill climbing PDBs (HC) are more informative (4 solved tasks) than other heuristics (0 or 1 solved task). Adding Cartesian abstractions to the hill climbing PDBs leads to worse heuristics. In principle, Scorpion should be able to produce better estimates given more heuristics, but having a larger set of heuristics can make finding a good order for saturated cost partitioning harder.

In the Data-Network domain it is beneficial to use Cartesian abstractions (14 solved tasks vs. 12 solved tasks without Cartesian abstractions), whereas all heuristic subsets solve almost the same number of tasks in both variants of Organic-Synthesis.

When using hill climbing PDB heuristics, Scorpion is unable to solve any Petri-Net-Alignment tasks within the given limits. This is the case since the hill climbing algorithm starts by computing a PDB for each goal variable and then calculates the maximal additive subsets of these PDBs. The latter step runs out of memory for all tasks from the Petri-Net-Alignment domain, because of the large number of goal facts.[2]

The Snake and Spider domains benefit from using PDB heuristics. With systematic or hill climbing PDBs Scorpion solves 13 and 15 tasks, in these two domains. Using only Cartesian abstractions leads to solving two fewer tasks in each of the two domains.

In the Termes domain hill climbing PDBs with at least one other type of heuristic solves 14 tasks, whereas the other configurations solve 12 or 13 tasks.

Acknowledgments

We would like to thank all Fast Downward contributors. Special thanks go to Álvaro Torralba and Vidal Alcázar for their h^2 mutexes code.

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 Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Alkhazraji, Y.; Wehrle, M.; Mattmüller, R.; and Helmert, M. 2012. A stubborn set algorithm for optimal planning. In De Raedt, L.; Bessiere, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P., eds., *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, 891–892. IOS Press.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, 1007–1012. AAAI Press.

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

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the most out of pattern databases for classical planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2357–2364. AAAI Press.

[2]All tasks in the Petri-Net-Alignment domain share the same problem file, which has 268 goal facts, and differ only in the domain files.

Seipp, J., and Helmert, M. 2018. Counterexample-guided Cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research* 62:535–577.

Seipp, J.; Keller, T.; and Helmert, M. 2017. Narrowing the gap between saturated and optimal cost partitioning for classical planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3651–3657. AAAI Press.

Seipp, J. 2018. *Counterexample-guided Cartesian Abstraction Refinement and Saturated Cost Partitioning for Optimal Classical Planning*. Ph.D. Dissertation, University of Basel.

Wehrle, M., and Helmert, M. 2014. Efficient stubborn sets: Generalized algorithms and selection strategies. In Chien, S.; Fern, A.; Ruml, W.; and Do, M., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 323–331. AAAI Press.