Odin: A Planner Based on Saturated Transition Cost Partitioning

Dominik Drexler, Jendrik Seipp, David Speck,

Linköping University, Linköping, Sweden
⟨dominik.drexler, jendrik.seipp, david.speck⟩@liu.se

This planner abstract describes an optimal classical planner called *Odin*. Odin uses A* search (Hart, Nilsson, and Raphael 1968) with an admissible heuristic (Pearl 1984) based on abstraction heuristics and saturated transition cost partitioning (Keller et al. 2016) to find optimal plans. Odin's main strength is in tasks where optimal plans contain the same actions multiple times, which is often the case in transportation domains.

Introduction

Odin is an optimal classical planner and implements a new version of our work on subset-saturated transition cost partitioning (Drexler, Speck, and Mattmüller 2020; Drexler, Seipp, and Speck 2021). The planner is built on top of Fast Downward (Helmert 2006) and Scorpion (Seipp and Helmert 2018) and runs A* (Hart, Nilsson, and Raphael 1968) with two different abstraction heuristics: 1. Cartesian abstraction heuristics of goal and landmark diversification (Seipp and Helmert 2018) with the batch refinement strategy of Speck and Seipp (2022) and 2. Pattern databases (Haslum et al. 2007) for systematic patterns for saturated cost partitioning.

Saturated Transition Cost Partitioning

Saturated transition cost partitioning is a method for admissibly combining the information of a collection of abstraction heuristics (Keller et al. 2016; Drexler, Seipp, and Speck 2021). Given an ordered set of abstraction heuristics, it assigns to each heuristic a fraction of the remaining costs to preserve its heuristic estimates, and leaves the remaining costs for subsequent heuristics.

The remaining costs are represented as a transition cost function $tcf : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ that maps state-operator pairs (or transitions) to real-valued costs. Since the number of transitions is exponential in the size of the input task, the performance of the method depends heavily on a compact representation of tcf. We use binary decision diagrams (BDDs) (Bryant 1986) from the CUDD library (Somenzi 2015) to compactly represent sets of states associated with the same cost value.

A special case of a transition cost function is an operator cost function $ocf : \mathcal{O} \to \mathbb{R}$, which maps operators to real-valued costs. Odin uses operator cost functions as a fallback solution when using transition cost functions is too time or memory intensive. More precisely, we use transition cost functions when the number of transitions in an abstraction heuristic is less than 40000.

The quality of a saturated cost partitioning heuristic depends heavily on the order of the heuristics. Therefore, saturated cost partitioning considers a collection of orders and evaluates its heuristic estimate to the maximum of the heuristic estimates over all orders. Since orders are optimized for a particular state, we compute new orders in an online manner during the search (Seipp 2021).

Keeping the heuristic estimates of all states is sometimes wasteful. Therefore, we preserve the heuristic estimates of only those states that are within a fixed perimeter of the goal. The perimeter is the distance from the initial state to a goal in the (Seipp and Helmert 2019) abstraction. To make subtraction of transition cost functions more efficient, we also subtract no costs for transitions to unsolvable states.

Operator Pruning Techniques

We use two operator pruning techniques, one as preprocessing and one during the search. As preprocessing, after grounding the input task, we run to apply h^2 preprocessor of Alcázar and Torralba (2015), which prunes spurious actions and simplifies the task. During the search we use strong stubborn set pruning (Alkhazraji et al. 2012; Wehrle and Helmert 2014; Röger et al. 2020). However, since the effectiveness of strong stubborn set pruning depends strongly on the domain and sometimes the computation does not pay off, we disabled pruning if the fraction of pruned states is less than 20% of the total successor states after 1000 expansions.

Configurations

Odin uses a different configuration depending on the PDDL language features that are present after grounding and simplifying the task.

No conditional effects and no axioms Odin runs saturated transition cost partitioning on CARTESIAN and SYS-SCP abstraction heuristics.

Conditional effects and no axioms Odin runs the Scorpion planner's saturated operator cost partitioning implementation on the SYS-SCP abstraction heuristic. In principle, Odin can be extended to work with conditional effects,

but it does not work out of the box because conditional effects introduce an additional level of state dependency.

Axioms Odin runs Dijkstra's algorithm (Dijkstra 1959).

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proc. ICAPS 2015*, 2–6.

Alkhazraji, Y.; Wehrle, M.; Mattmüller, R.; and Helmert, M. 2012. A Stubborn Set Algorithm for Optimal Planning. In *Proc. ECAI 2012*, 891–892.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8): 677–691.

Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1: 269–271.

Drexler, D.; Seipp, J.; and Speck, D. 2021. Subset-Saturated Transition Cost Partitioning. In *Proc. ICAPS 2021*, 131–139.

Drexler, D.; Speck, D.; and Mattmüller, R. 2020. Subset-Saturated Transition Cost Partitioning for Optimal Classical Planning. In *ICAPS Workshop on Heuristics and Search for Domain-independent Planning*, 23–31.

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 *Proc. AAAI 2007*, 1007–1012.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Keller, T.; Pommerening, F.; Seipp, J.; Geißer, F.; and Mattmüller, R. 2016. State-dependent Cost Partitionings for Cartesian Abstractions in Classical Planning. In *Proc. IJCAI 2016*, 3161–3169.

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

Röger, G.; Helmert, M.; Seipp, J.; and Sievers, S. 2020. An Atom-Centric Perspective on Stubborn Sets. In *Proc. SoCS 2020*, 57–65.

Seipp, J. 2021. Online Saturated Cost Partitioning for Classical Planning. In *Proc. ICAPS 2021*, 317–321.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *JAIR*, 62: 535–577.

Seipp, J.; and Helmert, M. 2019. Subset-Saturated Cost Partitioning for Optimal Classical Planning. In *Proc. ICAPS 2019*, 391–400.

Somenzi, F. 2015. CUDD: CU Decision Diagram Package – Release 3.0.0. https://github.com/ivmai/cudd. Accessed: 2023-02-20.

Speck, D.; and Seipp, J. 2022. New Refinement Strategies for Cartesian Abstractions. In *Proc. ICAPS 2022*, 348–352.

Wehrle, M.; and Helmert, M. 2014. Efficient Stubborn Sets: Generalized Algorithms and Selection Strategies. In *Proc. ICAPS 2014*, 323–331.