SYMPLE: Symbolic Planning based on EVMDDs

David Speck and **Florian Geißer** and **Robert Mattmüller**
University of Freiburg, Germany
{speckd, geisserf, mattmuel}@informatik.uni-freiburg.de

Abstract

SYMPLE is a classical planner which performs bidirectional symbolic search. Symbolic search has proven to be a useful approach to optimal classical planning and is usually based on Binary Decision Diagrams (BDDs). Our approach is based on an alternative data structure called Edge-valued Multi-valued Decision Diagrams (EVMDDs), which have some structural advantages over BDDs.

Introduction

The motivation for SYMPLE originates from two related sources. First, the observation that symbolic planning is a useful and powerful approach to optimal planning. Symbolic planners are similar to their explicit counterparts, but operate on entire sets of states instead of single states. Usually, decision diagrams are used as underlying symbolic data structure. The most popular decision diagrams are Binary Decision Diagrams (BDDs) (Bryant 1986) in symbolic search (Kissmann, Edelkamp, and Hoffmann 2014; Torralba et al. 2014). Second, that an alternative symbolic data structure, the so-called Edge-Multi-Valued Decision Diagrams (EVMDDs) (Lai, Pedram, and Vrudhula 1996; Ciardo and Siminiceanu 2002), were successfully used in planning with state-dependent action costs (Geißer, Keller, and Mattmüller 2015; 2016; Mattmüller et al. 2018). In BDD-based symbolic planning, each BDD represents a set of states. Multiple BDDs are required to encode at what cost the states are reachable. In contrast, EVMDDs can be used as underlying data structure to encode the costs of states in the same diagram which also encodes the reachability of states. Regarding planning tasks with diverse action costs, BDD-based approaches have to bucket over different costs (e.g. g-values), while our EVMDDs-based approach maintains a single decision diagram and can therefore be more compact.

SYMPLE and the underlying concept was original presented in Speck, Geißer, and Mattmüller (2018). The focus of our previous work was on the theory of EVMDD-based symbolic search for (optimal) planning. Representations of state sets, transition relations and new EVMDD operations required for EVMDD-A* were presented. While an empirical evaluation showed that BDD-A* is superior in many tasks with unit-costs, EVMDD-A* outperforms other approaches in domains with state-dependent costs. Unfortunately, the IPC 2018 has no track with state-dependent action costs, yet. Nevertheless, the compactness of SYMPLE can be an advantage over other symbolic approaches for planning tasks with diverse action costs.

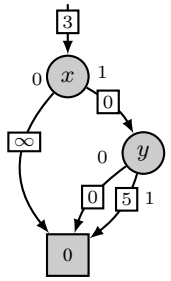
In this paper we will focus on the capabilities and implementation of SYMPLE, a bidirectional symbolic search planner based on EVMDDs. We give a short description of how SYMPLE represents states, costs and actions. EVMDD-A* is described, which is used for the actual search. In addition, the implementation of our planner is presented in detail. Finally, the differences between SYMPLE-1 and SYMPLE-2 are presented, which includes a new method of automated reformulating of planning tasks to simplify grounding.

Planning with EVMDDs

In this chapter we briefly describe how symbolic planning with EVMDDs can be realized.

EVMDD. A possible representations for functions of the form $f : \mathcal{S} \to \mathbb{Q} \cup \{\infty\}$ are *Edge-Valued Multi-Valued Decision Diagrams* (EVMDDs), where $\mathcal{S}$ denotes the set of factored states of a given planning task. An EVMDD is a rooted directed acyclic graph with a dangling incoming edge to the root node. Internal nodes correspond to variables v, and each node has $|\mathcal{D}_v|$ successors with an assigned weight to the edge, where $\mathcal{D}_v$ is the finite domain of variable v. A function can be evaluated by traversing the graph according to the variable assignment and simultaneously adding up the edge weights. The resulting sum is finally the function value for the corresponding variable assignment. An example is shown in Figure 1 where edge labels are written next to the edges and edge weights are written in boxes on the edges.

Symbolic Structures. Symbolic search operates on sets of states by performing operations. Here, states are represented as functions that map each state to the associated cost with which the state can be reached. Note, that a state s which is mapped to ∞ has infinity cost and thus is not reachable. For example, consider Figure 1. The EVMDD $\mathcal{E}$ represents the set of states $S = \{s | s(x) = 1\}$, since all other states are mapped to ∞. At the same time, the EVMDD encodes the cost of these states: $s_1 = \{x \doteq 1, y \doteq 0\}$ has a cost



$\mathbf{x}$	0	0	1	1
$\mathbf{y}$	0	1	0	1
$\mathbf{cost(x, y)}$	∞	∞	3	8

Figure 1: Left: An EVMDD $\mathcal{E}$ which represents a set of states and their costs. Right: The function represented by $\mathcal{E}$.

of 3 while $s_2 = \{x \doteq 1, y \doteq 1\}$ has a cost of 8. Similarly, actions can be represented as so called *transition relations*. Such transition relations are used to generate successor states and their costs which corresponds to applying actions in explicit planning. For more details we refer to (Speck, Geißer, and Mattmüller 2018).

EVMDD-A*. Once states and actions, both associated with costs, are represented as EVMDDs it is possible to start the actual search. Similar to explicit search, the main idea is to represent all promising states which might lead to the goal in an open list. The open list is a single EVMDD encoding the g-values (reachability costs) of each state. In each iteration, states with the smallest g-value are expanded. More specifically, all applicable actions represented as transition relations are applied to these states, resulting in new successor states. These states are again mapped to their g-value and added to the open list. As SYMPLE performs bidirectional search, separate open and closed lists for forward and backward search are maintained. A search step consists either of a backward or a forward search step (and modifies the respective open and closed lists). If a state of the current search is expanded and was already contained in the closed list of the search in the opposite direction, a goal path is found. Its cost is determined by adding the respective EVMDDs. If an optimal plan is desired, search has to continue, until it is proven that there is no cheaper goal path. Finally, plan reconstruction is executed for both directions and the returned plans are combined.

Implementation of SYMPLE

This chapter describes the technical aspects of the SYMPLE planner in detail. Furthermore, we describe the different configuration for each classical track of the IPC 2018. SYMPLE is build on top of the Fast Downward Planning System (Helmert 2006).

Preprocessing. SYMPLE's preprocessing is taken from GAMER (Kissmann, Edelkamp, and Hoffmann 2014) and SYMBA (Torralba et al. 2014), two former winners of IPCs (2008 and 2014). This includes the following procedures:

- GAMER's SAS$^+$ encoding (Kissmann, Edelkamp, and Hoffmann 2014)
- h^2 invariant computation and pruning of spurious actions (Alcázar and Torralba 2015)
- GAMER's and SYMBA's variable ordering (Kissmann and Edelkamp 2011)

Furthermore, in SYMPLE, we combine as many actions as possible into a transition relation, until the representation exceeds 100k nodes. Similarly, invariants are merged together and represented as EVMDDs in order to prune unreachable states.

Search. SYMPLE performs a bidirectional variant of EVMDD-A* (Speck, Geißer, and Mattmüller 2018) using the blind heuristic. At each iteration either a forward or backward search step is performed. In order to decide which direction appears to be more promising, the runtime of the last forward step is compared to the runtime of the last backward step. Conditional effects are encoded by extending the transition relations (Kissmann, Edelkamp, and Hoffmann 2014). To the authors' best knowledge, MEDDLY is currently the only decision diagram library supporting EVMDDs. Thus, the underlying library for EVMDD operations is an extended version of MEDDLY-0.14 (Babar and Miner 2010). The extension consists of operations necessary to realize symbolic planning (Speck, Geißer, and Mattmüller 2018) and the encoding of infinite costs. In order to save memory, we uses the "pessimistic" node deletion policy of MEDDLY, i.e. nodes are removed as soon as they become disconnected.

Track-Configurations. As SYMPLE was developed for optimal planning with state-dependent action costs using an A* variant, the main focus is on optimal planning. Nevertheless, optimal planners can usually be easily modified to participate at other tracks. SYMPLE participates in four different classical planning tracks at the IPC 2018: the optimal, the bounded-cost, the satisficing and the agile track. Generally it would be desirable to use different search techniques and heuristics tailored to the requirements of each track, but we have not yet studied these techniques for EVMDDs. In the following we describe the small changes made to SYMPLE to fit the requirements of the individual tracks.

- **All Tracks.** Bidirectional EVMDD with blind heuristic.
- **Optimal planning.** Once a plan is found, the search is continued until a cheaper plan is found or it is proven that no cheaper plan can exist.
- **Bounded-cost planning.** A plan found is only returned if it costs less than the specified bound.
- **Satisficing planning.** All plans found are returned. The search continues until an optimal plan has been found or the time has elapsed.
- **Agile planning.** As soon as a plan is found, it is returned.

SYMPLE-1 vs. SYMPLE-2

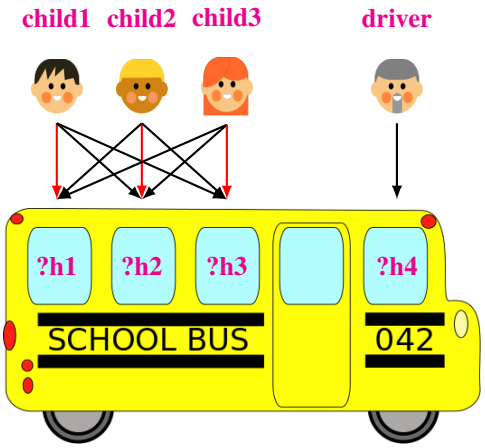
SYMPLE-2 differs from SYMPLE-1 only in the translation step. Both versions are based on the Fast Downward Planning System (Helmert 2006), and thus use the same translation unit to ground the lifted PDDL representation. By using expert knowledge, PDDL tasks are often reformulated beforehand, so that it is easy for planners to ground the planning task. In principle, such reformulations can be performed by the planner, and the IPC 2018 organizers announced that they plan to introduce tasks which are difficult to ground for current planners. SYMPLE-2 tackles the problem of the generation of duplicate redundant actions. Due to symmetries, grounded planning tasks may contain several identical actions that only differ in name, as the order of the action arguments does not affect precondition and effects. Detecting these symmetries and fixing the order of the arguments is the core addition of SYMPLE-2, which results in fewer redundant actions.

To illustrate this, consider a simple planning domain where the goal is to drive children to their school (Figure 2). The corresponding planning instance consists of four humans: one bus driver and three children. Without symmetry detection, grounding action `drive-to-school` results in six different actions: for parameter $?h4$, the only possible substitution is the bus driver, as he is the only one with a `license`. For parameters $?h1$, $?h2$ and $?h3$ however, all combinations of the three children are possible, which leads to $3! = 6$ grounded actions. It is easy to see that these actions are redundant, as they result in the exact same precondition, effect and cost. Although grounded actions can easily be checked for equivalence, this still implies that for n action parameters, an action similar to `drive-to-school` results in $n!$ generated actions and $\mathcal{O}(n!)$ equivalence checks. Our approach now reformulates the lifted PDDL action by introducing an ordering on the objects of symmetric action arguments. This ordering is only used during action generation and automatically discarded afterwards. Therefore, apart from omitted redundant actions, the resulting task is equivalent to a task where no symmetry detection is performed.

To identify such symmetries, we compute graph automorphisms of the induced planning graph of the lifted PDDL representation similar in spirit to Sievers et al. (2017), and Pochter, Zohar, and Rosenschein (2011). After symmetries in the action arguments are detected, we fix the order of these action arguments by introducing a predicate **succ**, as shown in Figure 2. The predicate **succ** is reflexive, since there may be actions where several parameters can be substituted by the same object constant (here: human). Note that the planning graph is not affected by the reformulation, as only redundant actions are discarded.

Acknowledgments

David Speck was supported by the German National Science Foundation (DFG) research unit FOR 1513 on Hybrid Reasoning for Intelligent Systems (`http://www.hybrid-reasoning.org`).




```
1 ;;;;;;;;;;;;;;;; Domain ;;;;;;;;;;;;;;;;
2 (define (domain bus)
3  (:types human location)
4  (:constants bus school - location)
5  (:predicates
6   (at ?l - location ?h - human)
7   (license ?h - human)
8   (succ ?h1 ?h2 - human))
9
10  (:action drive-to-school
11   :parameters (?h1 ?h2 ?h3 ?h4 - human)
12   :precondition (and (not (= ?h1 ?h2))
13    (not (= ?h1 ?h3)) (not (= ?h1 ?h4))
14    (not (= ?h2 ?h3)) (not (= ?h2 ?h4))
15    (not (= ?h3 ?h4))
16    (at bus ?h1) (at bus ?h2)
17    (at bus ?h3) (at bus ?h4)
18    (license ?h4)
19    (succ ?h1 ?h2) (succ ?h2 ?h3))
20    :effect (and (at school ?h1)
21    (at school ?h2) (at school ?h3)
22    (not (at bus ?h1)) (not (at bus ?h2))
23    (not (at bus ?h3)))))
24
25 ;;;;;;;;;;;;;;;; Problem ;;;;;;;;;;;;;;;;
26 (define (problem bus-3-1)
27  (:domain bus)
28  (:objects
29   driver child1 child2 child3 - human)
30  (:init (at bus driver)
31   (at bus child1)
32   (at bus child2)
33   (at bus child3)
34   (license driver)
35   (succ driver driver) (succ driver child1)
36   (succ driver child2) (succ driver child3)
37   (succ child1 child1) (succ child1 child2)
38   (succ child1 child3) (succ child2 child2)
39   (succ child2 child3) (succ child3 child3))
40  (:goal (and
41   (at school child1)
42   (at school child2)
43   (at school child3))))
```

Figure 2: A domain and instance description where six redundant action are generated by grounding the task. The **succ** predicate is automatically added. The reformulation of the task contains only one action.

References

Alcázar, V., and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press. Palo Alto, CA, USA.

Babar, J., and Miner, A. 2010. MEDDLY: Multi-terminal and edge-valued decision diagram library. In *Proceedings of the Seventh International Conference on the Quantitative Evaluation of Systems (QEST 2010)*, 195–196. IEEE Computer Society. Los Alamitos, CA, USA.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Ciardo, G., and Siminiceanu, R. 2002. Using edge-valued decision diagrams for symbolic generation of shortest paths. In Aagaard, M. D., and O'Leary, J. W., eds., *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design (FMCAD 2002)*, 256–273. Berlin, Heidelberg, Germany: Springer.

Geißer, F.; Keller, T.; and Mattmüller, R. 2015. Delete relaxations for planning with state-dependent action costs. In Yang, Q.; Kong, H.; and Wooldridge, M., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 1573–1579. AAAI Press. Palo Alto, CA, USA.

Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for planning with state-dependent action costs. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 140–148. AAAI Press. Palo Alto, CA, USA.

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

Kissmann, P., and Edelkamp, S. 2011. Improving Cost-Optimal Domain-Independent Symbolic Planning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press. Menlo Park, CA, USA.

Kissmann, P.; Edelkamp, S.; and Hoffmann, J. 2014. Gamer and Dynamic-Gamer: Symbolic search at IPC 2014. In *Eighth International Planning Competition (IPC 2014)*, 77–84.

Lai, Y.; Pedram, M.; and Vrudhula, S. B. K. 1996. Formal verification using edge-valued binary decision diagrams. *IEEE Transactions on Computers* 45(2):247–255.

Mattmüller, R.; Geißer, F.; Wright, B.; and Nebel, B. 2018. On the Relationship Between State-Dependent Action Costs and Conditional Effects in Planning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*. AAAI Press. Menlo Park, CA, USA.

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 Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 1004–1009. AAAI Press.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2017. Structural symmetries of the lifted representation of classical planning tasks. In *ICAPS 2017 Workshop on Heuristics and Search for Domain-independent Planning*, 67–74.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018. Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. Accepted.

Torralba, Á.; Alcázar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. SymBA*: A symbolic bidirectional A* planner. In *Eighth International Planning Competition (IPC 2014)*, 105–109.