Symbolic Search for Oversubscription Planning

David Speck[1] **and Michael Katz**[2]
[1] University of Freiburg, Freiburg im Breisgau, Germany
[2] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
speckd@informatik.uni-freiburg.de, michael.katz1@ibm.com

Abstract

The objective of optimal oversubscription planning is to find a plan that yields an end state with a maximum utility while keeping plan cost under a certain bound. In practice, the situation occurs whenever a large number of possible, often competing goals of varying value exist, or the resources are not sufficient to achieve all goals. In this paper, we investigate the use of symbolic search for optimal oversubscription planning. Specifically, we show how to apply symbolic forward search to oversubscription planning tasks and prove that our approach is sound, complete and optimal. An empirical analysis shows that our symbolic approach favorably competes with explicit state-space heuristic search, the current state of the art for oversubscription planning.

Introduction

The objective of classical planning is to find a plan, i.e., a sequence of actions with low cumulative cost that leads to a goal state. Goal states are usually specified by a goal formula, and most planning algorithms are designed to find a plan that satisfies the entire formula. In practice, however, there can be a large number of possible, often competing goals of varying value, and a system (e.g. Mars rover) might not be able to achieve all these goals with the available resources (e.g. battery power). In such scenarios, it is natural to consider a *utility* of a state instead and search for states that maximize the overall utility value. If the action costs and state utility values, i.e., solution cost and solution utility, are comparable, the problem of finding a plan is called *net-benefit planning* (van den Briel et al. 2004). If solution cost and utility are not comparable, the problem is called *oversubscription planning* (Smith 2004). In other words, oversubscription planning defines a utility function that maps states to the utility value associated with achieving these states, and the objective is to find a plan whose cost does not exceed a specified cost bound, while reaching a state that achieves a high utility. Both net-benefit and oversubscription planning are also known under the common name partial satisfaction planning.

Over the past decades, both classical and partial satisfaction planning have seen a variety of competitive approaches, with the most successful being the explicit state-space heuristic search (Bonet and Geffner 1999) and the symbolic search (Cimatti et al. 1997). For net-benefit in particular, symbolic branch-and-bound search was introduced (Edelkamp and Kissmann 2009; Kissmann 2012), solving a so-called planning with soft-goals problem. More precisely, instead of a general state utility function, a subset of variables, so-called soft goals, is specified, each associated with a certain utility value. An introduction of a compilation of soft-goals into classical planning (Keyder and Geffner 2009), has allowed to apply classical planners to solving net-benefit planning tasks. The underlying idea behind the compilation is that each net-benefit planning task can be reduced to finding a shortest path from a single source node to a single goal node in a graph under a single objective function.

For oversubscription planning, however, unlike for net-benefit planning, there is no simple reduction to classical planning with a single objective function. In fact, the complexity of several fragment of these two problems is different (Katz and Mirkis 2016). The first practically efficient approaches to optimally solving oversubscription planning applied an explicit branch-and-bound search with heuristics adapting the ideas of classical planning to oversubscription planning (Mirkis and Domshlak 2013, 2014; Domshlak and Mirkis 2015; Muller and Karpas 2018). Later, Katz et al. (2019a) introduced reformulations of oversubscription planning as classical planning with two cost functions, but without an utility function. The reformulations allowed to directly adapt classical planning heuristics to oversubscription planning by simply ignoring the secondary cost function, improving the efficiency of branch-and-bound search. Finally, Katz and Keyder (2019) show that these reformulations are useful for search as well, allowing switching to the more efficient A* search (Hart, Nilsson, and Raphael 1968), performed on the reformulated task. The search can then use the aforementioned adapted heuristics from classical planning (Katz and Keyder 2019). However, if such heuristics are used as-is, and not adapted to account for the secondary cost function, their informativeness is not sufficient to compensate for the time spent on the computation. As a result, the informed search is often outperformed by a simple blind search. If, however, the heuristics are adapted to account for the secondary cost function, the performance of the informed search improves significantly (Katz and Keyder 2019). Despite these advances, an exhaustive uninformed branch-and-bound search remains somewhat competitive (Katz et al. 2019a; Katz and Keyder 2019).

Another state space exploration technique that is especially well-suited for an exhaustive search is symbolic search (McMillan 1993). In contrast to explicit search, in which individual states are generated and expanded, symbolic search operates on sets of states. It uses efficient data structures, Binary Decision Diagrams (BDDs) (Bryant 1986) and Algebraic Decision Diagrams (ADDs) (Bahar et al. 1997) to represent and manipulate these sets of states. Recently, Eifler et al. (2020) applied symbolic search in the context of plan explanation, where plan utility is defined not in terms of states but as a set of plan properties, with the aim of providing an analysis that identifies dependencies between these plan properties. Nonetheless, to the best of our knowledge, symbolic search techniques have not yet been applied to oversubscription planning.

In this paper, we investigate the use of symbolic search for optimal oversubscription planning. In particular, we show how to apply symbolic forward search to oversubscription planning and introduce an algorithm that is sound, complete and optimal. Finally, an empirical study on various planning domains is conducted which shows that the presented algorithm results in an optimal planner that exceeds the current state of the art in terms of overall coverage.

Preliminaries

We consider oversubscription planning tasks (Smith 2004) that are characterized by the SAS^+ formalism (Bäckström and Nebel 1995).

Definition 1 (OSP Task). An *oversubscription planning (OSP) task* is a 6-tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, c, u, b \rangle$. $\mathcal{V}$ is a finite set of state variables, each associated with a finite domain $D_v = \{0, \dots, |D_v| - 1\}$. A fact is a pair (v, d), where $v \in \mathcal{V}$ and $d \in D_v$. For binary variables we also write v for $(v, 1)$ and $\neg v$ for $(v, 0)$. A partial variable assignment s over $\mathcal{V}$ is a consistent set of facts. If s assigns a value to each variable $v \in \mathcal{V}$, s is called a state. With $\mathcal{S}$ we refer to the set of all possible states defined over $\mathcal{V}$. $\mathcal{O}$ is a finite set of operators, where each operator is a pair $o = \langle pre_o, eff_o \rangle$ of partial variable assignments, called preconditions and effects. The state $s_0 \in \mathcal{S}$ is called the initial state. Each operator has non-negative cost defined by the function $c : \mathcal{O} \mapsto \mathbb{N}_0$. Finally, $u : \mathcal{S} \mapsto \mathbb{N}_0$ is an efficiently computable state utility function and $b \in \mathbb{N}_0$ specifies a cost bound.

An operator $o \in \mathcal{O}$ is applicable in a state s iff pre_o is satisfied in s, i.e., $pre_o \subseteq s$. Applying operator o to state s yields the state s' where $s'(v) = eff_o(v)$ for all variables $v \in \mathcal{V}$ for which eff_o is defined and $s'(v) = s(v)$ otherwise. A plan in oversubscription planning is defined as follows.

Definition 2 (Plan). A *plan* $\pi = \langle o_0, \dots, o_{n-1} \rangle$ for an oversubscription planning task Π is a sequence of operators applicable in s_0 that yield the states $s_1, \dots, s_n$, such that the cumulative operator cost $c(\pi) = \sum_{i=0}^{n-1} c(o_i)$ is smaller or equal to the cost bound b, i.e., $c(\pi) \leq b$. The utility $u(\pi)$ of plan π is defined by the utility of the end state s_n induced by π, i.e., $u(\pi) = u(s_n)$. Such a plan π is considered to be utility-optimal or simply optimal if there is no other plan π' such that $u(\pi') > u(\pi)$. A plan π is cheapest utility-optimal if it is cheapest among utility-optimal plans: there is no utility-optimal plan π' such that $c(\pi') < c(\pi)$.

Since the empty plan is already a valid plan for any OSP task, the main objective of oversubscription planning is to determine a plan with high utility. The search for an optimal plan, i.e., a plan with the highest utility, is called optimal oversubscription planning and is the main focus of this work.

Symbolic Search

Symbolic search is a technique for exploring state spaces that uses efficient data structures to represent and manipulate sets of states (McMillan 1993). Symbolic search algorithms resemble their explicit counterparts, but expand and generate whole sets of states in contrast to individual states. In symbolic planning, a set of states $S \subseteq \mathcal{S}$ is represented by its characteristic function χ_S, which is a Boolean function $\chi_S : \mathcal{S} \mapsto \{0, 1\}$. States contained in S are mapped to 1 and all other to 0, i.e., $\chi_S(s) = 1$ if $s \in S$ and $\chi_S(s) = 0$ otherwise. An operator $o \in \mathcal{O}$ can be represent as transition relation (TR) that is defined over sets of state pairs, namely predecessors and successors. A TR T_o representing an operator $o \in \mathcal{O}$ is a function $\chi_T : \mathcal{S} \times \mathcal{S}' \mapsto \{0, 1\}$ which maps the pairs of states (s, s') to 1 iff successor s' is reachable from predecessor s by applying operator $o \in \mathcal{O}$. Given a set of states S and a TR T_o, the $\mathrm{image}(S, T_o)$ operation computes all successors of S with respect to the operator o. Note that a single TR can in general represent multiple operators with the same cost (Torralba 2015). From now on, if it is clear from the context, we sometimes simplify notation and use the same symbol S for a set of states S and the corresponding characteristic function χ_S.

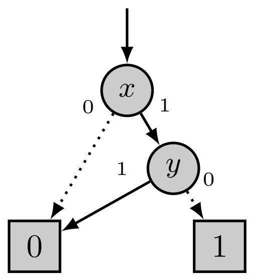
Symbolic representation of a planning task makes it possible to carry out symbolic forward search. The search starts with the characteristic function of the initial state $\chi_{\{s_0\}}$ and iteratively computes the successors until a function is found which meets a certain stopping condition. In classical planning, e.g., the stopping condition is an non-empty intersection with the goal. Finally, a plan reconstruction (Torralba 2015) is performed to obtain the final plan. This is necessary because, in contrast to explicit search, the predecessor states of an expanded state are not directly known in symbolic search. The plan is constructed by reconstructing the path from the initial state to the end state using a closed list which provides the perfect heuristic for all expanded states.

Symbolic Data Structures

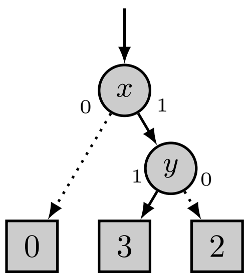
In symbolic search, characteristic functions are usually represented by compact and efficient symbolic data structures such as (reduced and ordered) Binary Decision Diagrams (BDDs) (Bryant 1986).

Definition 3 (Binary Decision Diagram). A BDD is a directed acyclic graph with a single root node and two terminal nodes: the 0-sink and the 1-sink. Each inner node corresponds to a binary[1] variable $v \in \mathcal{V}$ and has two successors,

[1]Each finite-domain variable $v \in \mathcal{V}$ can be represented by $\lceil \log_2 |D_v| \rceil$ binary variables.



(a) A BDD representing the characteristic function $\chi_S = x \wedge \neg y$.



(b) An ADD representing the numeric function $u = 2x + xy$.

Figure 1: Visualization of different decision diagrams.

where the low edge represents that variable v is false, while the high edge represents that variable v is true. By traversing the BDD according to a given assignment, the represented function can be evaluated.

Figure 1a visualizes the BDD which represents the characteristic function $\chi_S = x \wedge \neg y$, i.e., the set of states $S = \{s \in \mathcal{S} \mid s(x) = 1 \wedge s(y) = 0\}$.

Algebraic Decision Diagrams (ADDs) (Bahar et al. 1997) and Edge-Valued Multi-Valued Decision Diagrams (EVMDDs) (Lai, Pedram, and Vrudhula 1996; Ciardo and Siminiceanu 2002) have been successfully used to represent numerical functions (Hansen, Zhou, and Feng 2002; Torralba, Linares López, and Borrajo 2013; Speck, Geißer, and Mattmüller 2018a,b) in symbolic planning. In this work, we focus on ADDs in addition to BDDs which makes it possible to represent the utility function u as a decision diagram. The definition of an ADD is similar to the definition of BDDs, but using an arbitrary number of terminal nodes with different discrete values including real numbers. Figure 1b visualizes the ADD which represents the function $u = 2x + xy$. From now on we assume that numeric functions u are represented as ADD if not stated differently.

Operations. The $\max(u)$ operation determines the maximum value of a function u, i.e., $\max(u) = \max_{s \in \mathcal{S}} u(s)$. Note that the terminal nodes of an ADD u describe all possible values that u can have. This makes it possible to efficiently determine the maximum value of u once the ADD is created. The function shown in Figure 1b has exactly three different values, where 3 is the maximum value. In addition, an ADD can be disassembled into multiple BDDs, one for each terminal node, in polynomial time and memory with respect to the ADD size (number of nodes) by substituting terminal nodes (Torralba 2015). For example, the ADD shown in Figure 1b can be disassembled into three different BDDs, one for each terminal node. Figure 1a depicts the BDD representing all states for which the evaluation of function $u = 2x + xy$ is 2. The $\arg\max(u)$ operation takes advantage of this property and returns the set of states with the maximum value of a function u as BDD, i.e. $\arg\max(u) = \arg\max_{s \in \mathcal{S}} u(s)$. Finally, the intersection $u \wedge S$ between an ADD u and BDD S results in a new ADD u_S, where $u_S(s) = u(s)$ if $s \in S$ and $u_S(s) = -\infty$ otherwise. Intuitively, this operation preserves only the utility values of states in S which makes it for example possible to determine the maximum utility value of a given set of states S by applying $\max(S \wedge u)$.

Symbolic Search for OSP

In this section, we describe and explain how symbolic forward search can be applied to oversubscription planning and prove that our presented algorithm SYM-OSP is sound, complete and optimal. Finally, we discuss the possible application of symbolic backward and bidirectional search to oversubscription planning and the associated challenges and issues. For simplicity, we define SYM-OSP only explicitly for oversubscription planning tasks with unit costs, i.e., where each operator has an application cost of 1. Symbolic uniform search, also known as Dijkstra's algorithm (Dijkstra 1959) can be used for non-unit cost oversubscription planning tasks, which is possible by bucketing (disassembling) transition relations, open lists and closed lists into subsets with identical cost values represented as BDDs (Edelkamp and Kissmann 2009; Torralba 2015). Note that the presented algorithm and all presented concepts and theoretical results generalize to oversubscription planning tasks with non-unit operator cost.[2]

Algorithm SYM-OSP

The underlying idea of SYM-OSP (Algorithm 1) is to perform an exhaustive forward search and to explore all states reachable until the cost bound is reached or a state with maximal utility is found. Once one of the termination criteria is satisfied, the plan is reconstructed. Recall, that the plan reconstruction is necessary because, in contrast to explicit search, the predecessor states of an expanded state are not directly known in the symbolic search. SYM-OSP is analogous to symbolic forward search for classical planning, with the difference that in classical planning we terminate as soon as the open list is empty or a goal state is found.

Note that all sets of states are represented as BDDs and all numeric functions as ADDs. In detail, SYM-OSP (Algorithm 1) works as follows. First, the open list is initialized as the singleton set only containing the initial state, and the closed list is initialized as the empty set of states (line 1). The BDD π_S maintains the previously expanded states with the highest utility value found and is thus also initialized as the singleton set only containing the initial state (line 2). Similarly, with π_u we store the highest utility of all states that have been expanded so far and with π_g we store the cheapest cost with which a state with utility π_u can be reached. Recall that the empty plan is already a valid, but often not optimal plan for each OSP task. The algorithm continues as long as 1) the open list is not empty, 2) the cost bound has not been exceeded, and 3) no state with the highest possible utility has been found (line 4). In each iteration, the current set of states *open* is expanded and the already expanded states *closed* are removed from the resulting set of successors. In line 7 the maximum utility value of all new states *open* is

[2]Our implementation supports operators with non-unit costs.

Algorithm 1: SYM-OSP for *unit cost* OSP tasks

```
   Data: OSP Task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, \mathcal{O} \mapsto 1, u, b \rangle$
   Result: Cheapest utility-optimal plan $\pi$
1  $open, closed \leftarrow \{s_0\}, \emptyset$
2  $\pi_S, \pi_u, \pi_g \leftarrow \{s_0\}, u(s_0), 0$
3  $g \leftarrow 1$
4  while $open \neq \emptyset$ and $g \leq b$ and $\pi_u < \max(u)$ do
5      $closed \leftarrow closed \lor open$
6      $open \leftarrow (\bigcup_{o \in \mathcal{O}} \text{image}(open, T_o)) \land \neg closed$
7      if $\pi_u < \max(open \land u)$ then
8          $\pi_S \leftarrow \arg\max(open \land u)$
9          $\pi_u \leftarrow \max(open \land u)$
10         $\pi_g \leftarrow g$
11     $g \leftarrow g + 1$
12 return $\text{ReconstPlan}(\Pi, open, closed, \pi_S, \pi_u, \pi_g)$
```

computed and compared with the previously highest known utility value π_u. Lines 8 and 10 are reached when at least one state with higher utility has been found. Especially interesting is line 8, where the BDD π_S is updated with all the new states that yield the new and higher utility value π_u. Finally, the plan reconstruction (Torralba 2015) is performed to obtain and return the final plan (line 12).

Example 1. Consider a unit-cost oversubscription planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, \mathcal{O} \mapsto 1, u, b \rangle$ with two binary variables $\mathcal{V} = \{x, y\}$ where $D_x = D_y = \{0, 1\}$. Furthermore, Π has the initial state $s_0(x) = s_0(y) = 0$, the utility function $u = 2x + xy$ and the cost bound $b = 1$. There exist three operators $O = \{o_1, o_2, o_3\}$ where $o_1 = \langle \neg x \land \neg y, y \rangle$, $o_2 = \langle \neg x \land \neg y, x \rangle$ and $o_3 = \langle x \land \neg y, y \rangle$. The induced transition system of Π is depicted in Figure 2.

We apply SYM-OSP (Algorithm 1) to Π which starts with the BDD $open = \{s_0\} = \neg x \land \neg y$ representing a single state, namely the initial state (line 1). Note that Figure 1b visualizes the utility function u as ADD with a maximum utility of 3. The set of best states π_S is initialized with the initial state s_0, which can be reached with cost $\pi_g = 0$ and has the utility value of $u(s_0) = 0$ (line 2). The first expansion (line 6) leads to two new states $open = \{s_1, s_2\} = (\neg x \land y) \lor (x \land \neg y)$, both of which can be reached with cost $g = 1$. Since the intersection $open \land u$ maps the utility values of states $s \notin open$ to $-\infty$, the result of $\max(open \land u)$ is 2 (line 7 and 8). The operation $\arg\max(open \land u)$ returns the set of states with the maximum utility $\pi_S = \{s_2\} = x \land \neg y$ as BDD, which is shown in Figure 1a (line 8). Finally, the plan $\langle o_2 \rangle$, which leads to one of the state in π_S (here s_2), is reconstructed (line 12), since the cost bound of $b = 1$ is exceeded (line 4).

Theoretical Properties

In the following, we show that SYM-OSP is sound, complete and optimal for oversubscription planning.

Proposition 1. *Algorithm* SYM-OSP *is* sound *and* complete *for oversubscription planning.*

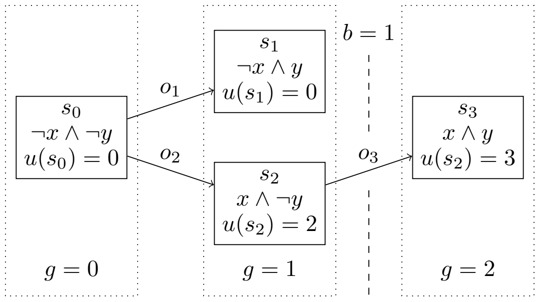


Figure 2: Visualization of the transition system induced by the oversubscription planning tasks Π in Example 1.

Proof. Any sequence of operators π that is applicable in the initial state and whose cumulative cost is lower than the cost bound b is a valid plan for an oversubscription planning task. Since SYM-OSP only expands states which are reachable within the cost bound (line 4), and all the resulting sequences of actions are applicable, SYM-OSP therefore returns only valid plans, proving that SYM-OSP is sound for oversubscription planning.

Algorithm SYM-OSP terminates if one of the following three cases occurs (line 4): 1) the entire reachable state space has been explored ($open = \emptyset$), 2) there are no more states that can be reached with costs less than the cost bound ($g > b$) or (3) a state with maximum utility value has been found ($\pi_u = \max(u)$). SYM-OSP expands all reachable states with increasing reachability cost g without considering already expanded states. Thus, at some point one of the condition holds and a valid plan is returned, proving that SYM-OSP is complete for oversubscription planning. □

It is possible to show that SYM-OSP is optimal, meaning that SYM-OSP returns utility-optimal plans (Definition 2) for a given oversubscription planning task Π.

Proposition 2. *Algorithm* SYM-OSP *is* optimal *for oversubscription planning.*

Proof. SYM-OSP only terminates when a state with the highest possible utility value is found or all reachable states are expanded within the given cost bound, in which case we have considered all relevant states and their utility values. Therefore, only plans with end states that have the highest reachable utility value are returned, proving that SYM-OSP is utility-optimal. □

Note that SYM-OSP not only finds an utility-optimal plan, but a *cheapest utility-optimal plans*, because SYM-OSP expands all reachable states with increasing costs. Finally, we would like to point out that all presented theoretical results also apply to symbolic uniform search, (Edelkamp and Kissmann 2009; Torralba et al. 2017), referred to from now on with SYM-OSP, that can be used to solve oversubscription planning with general operator costs.

Backward and Bidirectional Search

The dominant search strategy of modern symbolic planners (Torralba et al. 2014; Kissmann, Edelkamp, and Hoffmann

Algorithm	SYM-OSP		$A^\star_{\text{uADD}}$	$A^\star_{\text{mc}}$			BnB	
Benchmark Set (# Inst.)	**uBDD**	**uADD**	$\mathbf{h_{blind}}$	h_{blind}	$h^{\text{b}}_{\text{max}}$	$h^{\text{b}}_{\text{m\&s}}$	h_{blind}	$h^{\text{mc}}_{\text{lmcut}}$
25% BOUND (1667)	1271	**1274**	1165	1197	1190	1074	1183	1151
50% BOUND (1667)	990	**993**	860	901	902	828	893	867
75% BOUND (1667)	**866**	862	718	758	738	734	735	702
100% BOUND (1667)	**802**	793	629	668	655	676	643	618
OVERALL (6668)	**3929**	3922	3372	3524	3485	3312	3454	3338

Table 1: Coverage of the presented symbolic algorithms and heuristic search approaches on the oversubscription planning benchmark set which is based on the optimal track from the International Planning Competition 1998-2014 (Katz et al. 2019b). The best coverage for each benchmark set, i.e. the maximum entry of each row, is highlighted in bold.

2014; Speck, Mattmüller, and Nebel 2020) is bidirectional blind search (Torralba, Linares López, and Borrajo 2016; Speck, Geißer, and Mattmüller 2020). However, it is not straightforward to apply regression efficiently to oversubscription planning. In classical planning, the underlying idea of symbolic backward search, i.e., regression, is to start with the set of goal states and regress until the initial state is found. This is possible because the set of goal states is represented as a compact but often under-specified goal formula. However, since there is no goal formula in oversubscription planning, the starting point of the regression is not obvious. Since the objective is to find an end state with maximum utility value, it might be possible to partition all states according to their utility values and represent them as separate BDDs by disassembling the ADD representing the utility function. However, this would require to regress the entire state space by regressing several BDDs with multiple backward searches. All in all, it is not clear how to apply symbolic regression efficiently to oversubscription planning. We leave this task for future work.

Empirical Evaluation

The presented symbolic approaches to oversubscription planning were implemented[3] in the SYMBA (Torralba et al. 2014) planner, built on top of the FAST DOWNWARD planning system (Helmert 2006). The two approaches, SYM-OSP uBDD and SYM-OSP uADD, perform symbolic forward search (see Algorithm 1), with the difference being in the way the utility of a set of states is evaluated. SYM-OSP uBDD disassembles the utility function into multiple BDDs and performs multiple intersections to determine the maximum utility of a set of states, while SYM-OSP uADD uses a single ADD. In practice, ADDs are often better suited to determine the value of a single state in an explicit search, while a collection of BDDs is considered more suitable for symbolic search (Torralba 2015). In addition, we implemented $A^\star_{\text{uADD}}$ (Hart, Nilsson, and Raphael 1968) as an explicit search, representing the utility function as an ADD to determine the utility values of a single state. Algorithm $A^\star_{\text{uADD}}$ with the blind heuristic h_{blind} is the explicit counterpart of the algorithms SYM-OSP uBDD and SYM-OSP uADD.

[3]Available online: https://github.com/speckdavid/symbolic-osp

Setup

All experiments are conducted on the oversubscription planning benchmark set, based on the the optimal track from the International Planning Competition (IPC) 1998-2014 (Katz et al. 2019b), where goal facts are replaced with utilities. In total, the benchmark set consists of four parts, each consisting of the same 57 domains. The difference between the four parts is the cost bound, which for each planning instance is set at 25%, 50%, 75% or 100% of the cost of the optimal or best known solution respectively. Note that several of the domains contain operators with non-unit costs.

For comparison, publicly available planners are chosen that support PDDL (McDermott 2000) and are state of the art in oversubscription planning: branch-and-bound search BnB (Katz et al. 2019a) and $A^\star_{\text{mc}}$ search with multiple cost functions (mc) (Katz and Keyder 2019). Both approaches are based on reformulations that result in planning problems with multiple separate cost functions, but no utility function. The branch-and-bound search BnB can use two separate heuristics, one for guidance, i.e., to decide which node to expand next, and one for pruning nodes. Such heuristics h^{mc} are computed on the reformulated planning problem with multiple cost functions, while BnB is performed on the original problem. We compare our approach against BnB with the blind heuristic for guidance and the blind heuristic h_{blind} and the LM-cut heuristic $h^{\text{mc}}_{\text{lmcut}}$ (Helmert and Domshlak 2009) for pruning, as proposed by Katz et al. (2019a). Note that the known best performing configuration uses the blind heuristic h_{blind} for both heuristics (Katz et al. 2019a). $A^\star_{\text{mc}}$ search maintains multiple cost values for each state, one for each cost function. In addition, $A^\star_{\text{mc}}$ search for oversubscription planning can utilize bound-sensitive heuristics which are able to reason about the primary cost of a solution while taking into account secondary cost functions and bounds (Katz and Keyder 2019). We compare with $A^\star_{\text{mc}}$ search using the blind heuristic h_{blind}, the bound-sensitive max heuristic $h^{\text{b}}_{\text{max}}$ (Bonet and Geffner 1999) and the bound sensitive merge-and-shrink heuristic $h^{\text{b}}_{\text{m\&s}}$ (Helmert et al. 2014) introduced in Katz and Keyder (2019).

All experiments are run on a compute cluster with nodes equipped with two Intel Xeon Gold 6242 32-core CPUs, 20 MB cache and 188 GB shared RAM running Ubuntu 18.04 LTS 64 bit. The planners are build with 64 bit and run with a time limit of 30 minutes and memory limit of 4 GB.

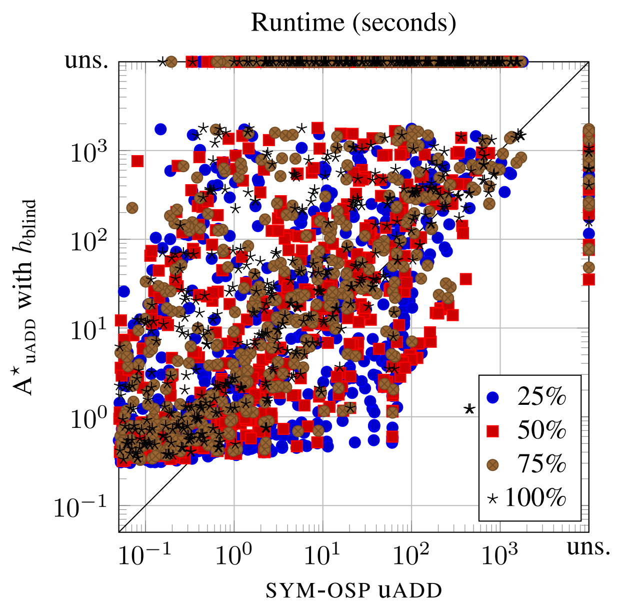


Figure 3: Runtime comparison of symbolic search and explicit search on the oversubscription planning benchmark set which is based on the optimal track from the IPC 1998-2014. With *uns.* we refer to instances that were not solved within the time and memory limits.

		SYM-OSP		A$^\star_{\text{uADD}}$	A$^\star_{\text{mc}}$			BnB	
		uBDD	**uADD**	$\mathbf{h_{blind}}$	h_{blind}	$h^{\text{b}}_{\text{max}}$	$h^{\text{b}}_{\text{m\&s}}$	h_{blind}	$h^{\text{mc}}_{\text{lmcut}}$
SYM-OSP	**uBDD**	-	**13**	**116**	**104**	**103**	**125**	**114**	**126**
	uADD	**13**	-	**116**	**102**	**103**	**125**	**114**	**124**
A$^\star_{\text{uADD}}$	$\mathbf{h_{blind}}$	26	28	-	0	26	46	15	**35**
A$^\star_{\text{mc}}$	h_{blind}	47	47	**47**	-	47	**62**	**48**	**82**
	$h^{\text{b}}_{\text{max}}$	53	51	**86**	**56**	-	**76**	**75**	**91**
	$h^{\text{b}}_{\text{m\&s}}$	38	40	**76**	43	44	-	**65**	**82**
BnB	h_{blind}	36	35	**48**	0	37	53	-	**47**
	$h^{\text{mc}}_{\text{lmcut}}$	26	25	25	0	6	43	4	-

Table 2: Per-domain coverage comparison of different algorithms on the complete oversubscription planning benchmark set (228 domains) which is based on the optimal track from the IPC 1998-2014. The entry in row r and column c shows the number of domains in which algorithm r solves more tasks than algorithm c. For each pair of algorithms we highlight the maximum of entries (r, c) and (c, r) in bold.

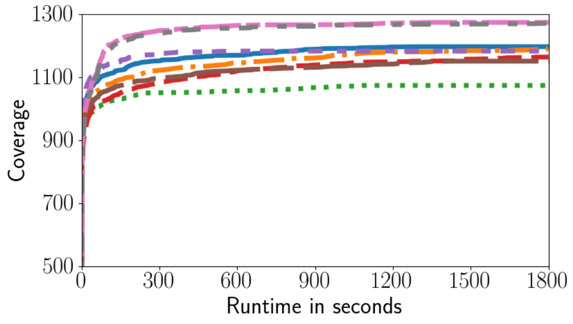
Experiments

Table 1 shows the coverage (sum of solved instances) of the presented symbolic approaches compared to explicit heuristic search approaches. Overall, as expected, the performance decreases with increasing cost bounds, since a greater cost bound generally requires more states to be considered and expanded to find a plan and prove its optimality in oversubscription planning. Next, we analyze and discuss the presented symbolic approaches in detail before comparing them to the state-of-the-art approaches that are based on an explicit heuristic search. Finally, we perform a per-domain comparison of all approaches.

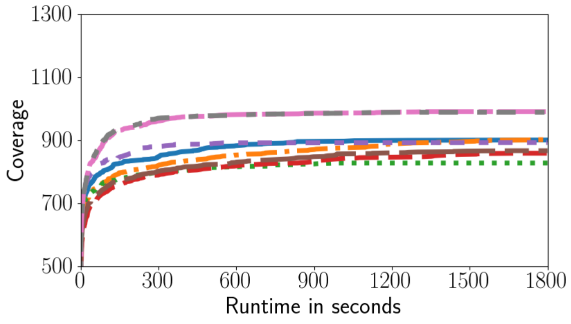
Symbolic approaches. Comparing the approaches using symbolic data structures, we can observe that symbolic search with a disassembled utility function in multiple BDDs (SYM-OSP uBDD) works best overall. The configuration SYM-OSP uADD, which represents the utility function as a single ADD, performs only slightly worse overall and better in the benchmark sets with smaller cost bounds (25% and 50%). The main advantage of evaluating the utility values of the open list with BDDs compared to ADDs is that the decision diagram library CUDD (Somenzi 2015) uses techniques such as *complement edges* to store BDDs more compactly (Brace, Rudell, and Bryant 1990). However, complement edges are not used to represent ADDs. Therefore, when the utility function is represented as an ADD, it is necessary to transform the open list represented as a BDD with complement edges into an ADD (without complement edges). Depending on the structure of the BDD, this leads to an overhead which can be avoided when representing the utility function as multiple BDDs. This is consistent with the literature, where representation of numeric functions as BDDs was shown in general to be more suitable for symbolic search (Torralba 2015).

Observe that the time required to create the utility function as ADD is negligible in most instances. In over 6300 of 6668 instances it takes less than one second to create the ADD, with a maximum in the remaining instances of about two minutes. Also, the time for disassembling the ADD into several BDDs (which is only carried out by SYM-OSP uBDD) is not a bottleneck in most instances, because in over 5700 instances this process takes less than one second. However, in 60 instances this procedure lasted more than two minutes.

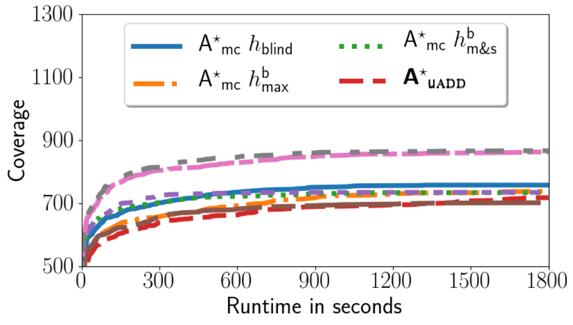
A comparison of symbolic forward search to explicit search, in which the utility function is represented symbolically, i.e., as ADD, reveals a performance gap. Figure 3 compares the run times of SYM-OSP uADD and its explicit counter part A$^\star_{\text{uADD}}$ with the blind heuristic h_{blind}. Note that symbolic blind search performs better overall, especially with increasing bounds, where the compact representation gets more important. Most instances of the 100% benchmark set are solved faster by symbolic forward search than by explicit forward search. In a few instances, however, explicit blind search outperforms symbolic blind search. Apart from implementation reasons, this has the explanation that in order to carry out symbolic search, all necessary symbolic data structures, such as the transition relations, have to be created first, which in some cases does not pay off in terms of time and memory. However, the dominance of symbolic search can be traced back to theoretical properties of BDDs, such as the fact that the number of nodes required to represent a given set of states, is at most linear in the number of states and variables. Further, it is possible to represent exponentially many states (in the number of variables) with only a polynomial number of nodes. For these reasons, symbolic search is well suited for exhaustive state space exploration.



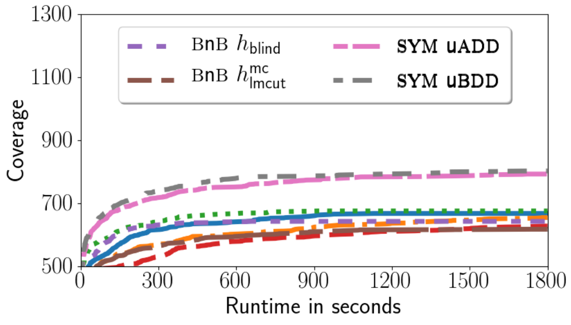
(a) 25% BOUND Benchmark set



(b) 50% BOUND Benchmark set



(c) 75% BOUND Benchmark set



(d) 100% BOUND Benchmark set

Figure 4: Coverage over time of the presented symbolic algorithms in comparison to heuristic search approaches on the oversubscription planning benchmark set which is based on the optimal track from the IPC 1998-2014.

Explicit heuristic approaches. A comparison between symbolic forward search for oversubscription planning and heuristic search algorithms based on reformulations shows an advantage of symbolic search in terms of overall coverage (Table 1). Here as well, the difference in performance between symbolic search and explicit heuristic search increases with an increasing bound and thus increasing plan length. The natural explanation is that heuristics do not pay off, which is also why blind search performs best and, as mentioned above, symbolic blind search is in practice often preferable to explicit blind search. This observation is underlined by Figure 4, which depicts the number of solved instances over time. While symbolic search solves fewer instances in the first seconds, after a short time it dominates all other approaches, i.e., it solves more instances overall and keeps this performance gap over time. Especially for greater cost bounds, the advantage in overall coverage tends to increase over time (Figures 4c and 4d), while it remains the same for smaller cost bounds (Figures 4a and 4b).

Per-domain comparison. Table 2 shows a per-domain comparison of the different approaches to oversubscription planning. As in classical planning, symbolic and explicit approaches shine in different domains. However, there is a clear advantage to symbolic search over explicit heuristic search overall. Symbolic search solves more instances in more than twice as many domains than any explicit search approach. Nevertheless, there are domains where explicit heuristic search performs better than symbolic search. This shows that, similar to classical planning, a potential portfolio planner of these complementary search strategies could result in a state-of-the-art planner that combines both strengths (Sievers et al. 2019).

Conclusion

We presented a novel approach to optimal oversubscription planning, SYM-OSP, based on symbolic forward search and proved that SYM-OSP is sound, complete and optimal. Our empirical evaluation shows that SYM-OSP performs better than other state-of-the-art approaches, for all tested cost bounds. In addition, SYM-OSP scales to larger cost bounds better than other approaches, making SYM-OSP suitable regardless of the cost bound. However, as usual, there are domains, where explicit heuristic search performs better than symbolic search.

For future work, we would like to investigate symbolic backward search for oversubscription planning, a necessary condition for both performing bidirectional symbolic search and a possible application of symbolic abstraction heuristics (Edelkamp 2002) to oversubscription planning. Finally, applying symbolic branch-and-bound search (Edelkamp and Kissmann 2009; Kissmann 2012) to oversubscription planning is an interesting research direction.

Acknowledgments

David Speck was supported by the German Research Foundation (DFG) as part of the project EPSDAC (MA 7790/1-1). We thank the anonymous reviewers for their comments and suggestions.

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS$^+$ Planning. *Computational Intelligence* 11(4): 625–655.

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1997. Algebraic Decision Diagrams and Their Applications. *Formal Methods in System Design* 10(2–3): 171–206.

Bonet, B.; and Geffner, H. 1999. Planning as Heuristic Search: New Results. In Biundo, S.; and Fox, M., eds., *Recent Advances in AI Planning. 5th European Conference on Planning (ECP 1999)*, volume 1809 of *Lecture Notes in Artificial Intelligence*, 360–372. Heidelberg: Springer-Verlag.

Brace, K. S.; Rudell, R. L.; and Bryant, R. E. 1990. Efficient implementation of a BDD package. In Smith, R. C., ed., *Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC 1990)*, 40–45.

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.; and O'Leary, J. W., eds., *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design (FMCAD 2002)*, volume 2517 of *Lecture Notes in Computer Science*, 256–273. Springer-Verlag.

Cimatti, A.; Giunchiglia, F.; Giunchiglia, E.; and Traverso, P. 1997. Planning via Model Checking: A Decision Procedure for *AR*. In Steel, S.; and Alami, R., eds., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP 1997)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, 130–142. Springer-Verlag.

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

Domshlak, C.; and Mirkis, V. 2015. Deterministic Oversubscription Planning as Heuristic Search: Abstractions and Reformulations. *Journal of Artificial Intelligence Research* 52: 97–169.

Edelkamp, S. 2002. Symbolic Pattern Databases in Heuristic Search Planning. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, 274–283. AAAI Press.

Edelkamp, S.; and Kissmann, P. 2009. Optimal Symbolic Planning with Action Costs and Preferences. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1690–1695. AAAI Press.

Eifler, R.; Steinmetz, M.; Torralba, Á.; and Hoffmann, J. 2020. Plan-Space Explanation via Plan-Property Dependencies: Faster Algorithms & More Powerful Properties. In Bessiere, C., ed., *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, 4091–4097. IJCAI.

Hansen, E. A.; Zhou, R.; and Feng, Z. 2002. Symbolic Heuristic Search Using Decision Diagrams. In Koenig, S.; and Holte, R. C., eds., *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation (SARA 2002)*, volume 2371 of *Lecture Notes in Artificial Intelligence*, 83–98. Springer-Verlag.

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.

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

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 Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM* 61(3): 16:1–63.

Katz, M.; and Keyder, E. 2019. A* Search and Bound-Sensitive Heuristics for Oversubscription Planning. In *ICAPS 2019 Workshop on Heuristics and Search for Domain-independent Planning (HSDIP)*.

Katz, M.; Keyder, E.; Pommerening, F.; and Winterer, D. 2019a. Oversubscription Planning as Classical Planning with Multiple Cost Functions. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 237–245. AAAI Press.

Katz, M.; Keyder, E.; Pommerening, F.; and Winterer, D. 2019b. PDDL benchmarks for oversubscription planning. https://doi.org/10.5281/zenodo.2576024.

Katz, M.; and Mirkis, V. 2016. In Search of Tractability for Partial Satisfaction Planning. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3154–3160. AAAI Press.

Keyder, E.; and Geffner, H. 2009. Soft Goals Can Be Compiled Away. *Journal of Artificial Intelligence Research* 36: 547–556.

Kissmann, P. 2012. *Symbolic Search in Planning and General Game Playing*. Ph.D. thesis, University of Bremen.

Kissmann, P.; Edelkamp, S.; and Hoffmann, J. 2014. Gamer and Dynamic-Gamer – Symbolic Search at IPC 2014. In *Eighth International Planning Competition (IPC-8): planner abstracts*, 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.

McDermott, D. 2000. The 1998 AI Planning Systems Competition. *AI Magazine* 21(2): 35–55.

McMillan, K. L. 1993. *Symbolic Model Checking*. Kluwer Academic Publishers.

Mirkis, V.; and Domshlak, C. 2013. Abstractions for Oversubscription Planning. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 153–161. AAAI Press.

Mirkis, V.; and Domshlak, C. 2014. Landmarks in Oversubscription Planning. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 633–638. IOS Press.

Muller, D.; and Karpas, E. 2018. Value Driven Landmarks for Oversubscription Planning. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 171–179. AAAI Press.

Sievers, S.; Katz, M.; Sohrabi, S.; Samulowitz, H.; and Ferber, P. 2019. Deep Learning for Cost-Optimal Planning: Task-Dependent Planner Selection. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019)*, 7715–7723. AAAI Press.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 393–401. AAAI Press.

Somenzi, F. 2015. CUDD: CU decision diagram package - release 3.0.0. https://github.com/ivmai/cudd. Accessed: 2020-02-20.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018a. Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 250–258. AAAI Press.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018b. SYMPLE: Symbolic Planning based on EVMDDs. In *Ninth International Planning Competition (IPC-9): planner abstracts*, 91–94.

Speck, D.; Geißer, F.; and Mattmüller, R. 2020. When Perfect Is Not Good Enough: On the Search Behaviour of Symbolic Heuristic Search. In Beck, J. C.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)*, 263–271. AAAI Press.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In Conitzer, V.; and Sha, F., eds., *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*, 9967–9974. AAAI Press.

Torralba, Á. 2015. *Symbolic Search and Abstraction Heuristics for Cost-Optimal Planning*. Ph.D. thesis, Universidad Carlos III de Madrid.

Torralba, Á.; Alcázar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. SymBA*: A Symbolic Bidirectional A* Planner. In *Eighth International Planning Competition (IPC-8): planner abstracts*, 105–109.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient Symbolic Search for Cost-optimal Planning. *Artificial Intelligence* 242: 52–79.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2013. Symbolic Merge-and-Shrink for Cost-Optimal Planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2394–2400. AAAI Press.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2016. Abstraction Heuristics for Symbolic Bidirectional Search. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3272–3278. AAAI Press.

van den Briel, M.; Sanchez, R.; Do, M. B.; and Kambhampati, S. 2004. Effective Approaches for Partial Satisfaction (Over-Subscription) Planning. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, 562–569. AAAI Press.