Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Loopless Top-K Planning

Julian von Tschammer,[1] Robert Mattmüller,[1] David Speck[1,2]

[1]University of Freiburg, Freiburg, Germany
[2]Linköping University, Linköping, Sweden
julian.vt@posteo.de, {mattmuel,speckd}@informatik.uni-freiburg.de

Abstract

In top-k planning, the objective is to determine a set of k cheapest plans that provide several good alternatives to choose from. Such a solution set often contains plans that visit at least one state more than once. Depending on the application, plans with such loops are of little importance because they are dominated by a loopless representative and can prevent more meaningful plans from being found.

In this paper, we motivate and introduce *loopless top-k planning*. We show how to enhance the state-of-the-art symbolic top-k planner, SYM-K, to obtain an efficient, sound and complete algorithm for loopless top-k planning. An empirical evaluation shows that our proposed approach has a higher k-coverage than a generate-and-test approach that uses an ordinary top-k planner, which we show to be incomplete in the presence of zero-cost loops.

Introduction

A prominent problem in graph theory is the k shortest paths problem with the objective of finding $k \in \mathbb{N}$ shortest distinct paths between two specific nodes in a graph (Bock, Kanter, and Haynes 1957). Considering the k shortest paths of a given graph, it turns out that there are often paths in the solution set that contain loops visiting at least one node more than once. Depending on the application, e. g., routing problems in a road network, paths containing loops are of little importance. Any path containing a loop is dominated by a loopless representative path p (often called a simple path), where all nodes visited by p are distinct. In practice, considering all paths may prevent finding more useful paths. Therefore two different versions of the k shortest paths problem exist: one that allows only loopless paths (Yen 1971; Hershberger, Maxel, and Suri 2007) and one that allows paths with and without loops (Eppstein 1998; Aljazzar and Leue 2011).

The planning counterpart of the k shortest paths problem is the top-k planning problem (Riabov, Sohrabi, and Udrea 2014) with the objective of finding k cost-optimal plans for a given planning task. Unlike the k shortest paths problem in graph theory, the top-k planning problem has so far been considered only in its basic form, which admits plans with loops, i. e., plans that may visit a state more than once (Katz et al. 2018; Speck, Mattmüller, and Nebel 2020). In certain planning scenarios, it is important to consider all plans, including plans with loops, as ordinary top-k planners do. For example, to provide all possibilities when user preferences are not fully known (Nguyen et al. 2012), to find a plan that satisfies complex (LTL) constraints (Gerevini et al. 2009), or when searching for an actual plan in a simplified version of a given problem, e. g., in an overapproximated or abstracted model representation (Seipp and Helmert 2018; Höller 2021), where not considering plans with loops may lead to an incomplete search.

However, in line with the arguments in graph theory for considering only loopless paths, in many planning applications, plans with loops are of little importance. Two prominent examples are the airport domain (Trüg, Hoffmann, and Nebel 2004) and the elevator domain (Koehler and Schuster 2000). In the airport domain, the ground traffic of airplanes must be coordinated. A set of best plans in this domain often includes plans where a single airplane is moved from its starting position to a free runway, then sent back to its starting position and coordinated back to the same runway, with no other airplane movements in between. While in the airport domain all operators have non-zero costs and thus all plans are enumerated at some point, these dominated plans with loops create a significant overhead and distract from actually interesting plans. In the elevator domain, one can observe an even worse effect when considering all plans. Here, only moving an elevator has a non-zero cost, whereas a passenger boarding or leaving an elevator has a cost of zero. This results in an infinite number of optimal plans, simply by a passenger repeatedly boarding and leaving an elevator. In such a case, the usefulness of an ordinary top-k planner is limited, as it gets stuck in generating optimal plans with zero-cost loops, which hinders the generation of more meaningful plans with potentially higher costs.

In this paper, we define loopless top-k planning and propose two different approaches to it. The first one is the generate-and-test approach SYM-K-GNT that uses an ordinary top-k planner. We show that SYM-K-GNT is sound in general but not complete in the presence of zero-cost loops. The second approach enhances the symbolic top-k planner SYM-K (Speck, Mattmüller, and Nebel 2020) yielding the generally sound and complete algorithm SYM-K-LL. An empirical evaluation shows that SYM-K-LL is overall more efficient and leads to a higher k-coverage than SYM-K-GNT.

Preliminaries

We consider classical planning tasks in the SAS^+ formalism (Bäckström and Nebel 1995) defined as follows:

Definition 1 (Planning Task)**.** *A* planning task *is a tuple* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$*.* $\mathcal{V}$ *is a finite set of* state variables $v \in \mathcal{V}$*, each associated with a finite* domain $\mathcal{D}_v$*. A* partial state s *is a mapping of a subset of variables* $\mathrm{vars}(s) \subseteq \mathcal{V}$ *to values in their domains. If* s *assigns a value* $s(v) \in \mathcal{D}_v$ *to each variable* $v \in \mathcal{V}$*, i.e,* $\mathrm{vars}(s) = \mathcal{V}$*,* s *is called a* state*.* $\mathcal{S}$ *denotes the set of all possible states over* $\mathcal{V}$*.* $\mathcal{O}$ *is a finite set of* operators *or* actions*. An operator* $o = \langle \mathrm{pre}_o, \mathrm{eff}_o \rangle \in \mathcal{O}$ *consists of the partial states* pre_o *and* eff_o *called* preconditions *and* effects*, respectively. An operator* $o \in \mathcal{O}$ *is* applicable *in a state* s *iff* $\mathrm{pre}_o \subseteq s$*. Applying* o *in* s *results in state* $s[o]$ *with* $s[o](v) = \mathrm{eff}_o(v)$ *if* $v \in \mathrm{vars}(\mathrm{eff}_o)$ *and* $s[o](v) = s(v)$ *otherwise. Each operator* $o \in \mathcal{O}$ *has a cost* $\mathrm{cost}(o) \in \mathbb{N}_0$*. Finally,* $s_0 \in \mathcal{S}$ *is the* initial state *and the partial state* $s_\star$ *is the* goal condition *specifying the set of all possible* goal states $S_\star \subseteq \mathcal{S}$*.*

The objective of classical planning is to determine plans that lead from the initial state to a goal state.

Definition 2 (Plan)**.** *A* plan $\pi = \langle o_0, \ldots, o_{n-1} \rangle$ *for a planning task* Π *is a sequence of applicable operators that generates a sequence of states* $\mathrm{states}(\pi) = \langle s_0, \ldots, s_n \rangle$*, where* $s_{i+1} = s_i[o_i]$ *for* $i \in \{0, \ldots, n-1\}$ *and* $s_n \in S_\star$*. The cost of a plan* π *is the sum of its operator costs, i. e.,* $\mathrm{cost}(\pi) = \sum_{i=0}^{n-1} \mathrm{cost}(o_i)$*. A plan* π *is called* loopless *if all states of* $\mathrm{states}(\pi)$ *are distinct, and* loopy *otherwise. With* P_Π *we refer to the (possibly infinite) set of all plans and with* $P_\Pi^{\ell\ell}$ *we refer to the finite set of all loopless plans for a planning task* Π*.*

Symbolic Search

In this paper, we will use and enhance SYM-K, a planner based on symbolic search. We briefly motivate and explain the relevant concepts related to symbolic search so that the modifications and enhancements presented can be understood. For a detailed introduction and summary of symbolic search in planning, we refer the reader to Torralba (2015) and Torralba et al. (2017), and for a comprehensive explanation of symbolic search for top-k planning, we refer the reader to Speck, Mattmüller, and Nebel (2020).

Symbolic search is a state space exploration technique that originated in model checking (McMillan 1993), in which entire sets of states $S \subseteq \mathcal{S}$ are expanded at once using decision diagrams, such as Binary Decision Diagrams (BDDs) (Bryant 1986), for concise representation. Similarly, decision diagrams are used to represent the transition relations (TRs) induced by operators $o \in \mathcal{O}$, i. e., the sets of state pairs (s, s') such that s' is the successor of s when o is applied in s. Given BDDs representing a set of states S and a TR T_o, the operation $\mathrm{image}/\mathrm{preimage}$ computes a BDD representing the set of successor/predecessor states S' of S through T_o. An advantage of symbolic search over explicit state space search is that it easily supports forward, backward, and bidirectional search in planning because it inherently supports sets of states that can occur when regressing a state with an operator.

In forward direction, the search starts with the singleton set containing the initial state and progresses the most promising set of states (lowest reachability costs) until a goal state is found. In backward direction, the search starts with the set of goal states and regresses the most promising set of states until the initial state is found. The best search strategy of modern symbolic planners (Torralba et al. 2014; Kissmann, Edelkamp, and Hoffmann 2014; Speck, Geißer, and Mattmüller 2018) is bidirectional search without heuristics (Torralba et al. 2017; Speck, Geißer, and Mattmüller 2020). Symbolic bidirectional search performs a separate symbolic forward and backward search and switches between the two search directions until the search frontiers meet and the solution found is proven to be optimal.

Finally, a plan reconstruction is performed to obtain the final plan (Torralba 2015). In symbolic search, the parents (predecessors in forward search; successors in backward search) of a state are not known directly, but all reachable states are stored together with their reachability costs in a closed list that forms the set of potential parent states. This makes it possible to perform the plan reconstruction as a greedy search opposed to the actual search direction, which is guided by the reachability cost of the stored states.

Speck, Mattmüller, and Nebel (2020) show that three modifications of ordinary symbolic search lead to a sound and complete algorithm for top-k planning: 1) the plan reconstruction is performed exhaustively instead of only reconstructing a single plan from the initial state to found goal states, 2) all newly generated states with their corresponding reachability costs are added to the open list without filtering out already expanded states[1], 3) and the termination criterion is adjusted so that the algorithm terminates when either k plans have been found or no more plans can be found.

Loopless Top-k Planning

We define loopless top-k planning as follows.

Definition 3 (Loopless Top-k Planning)**.** *Given a planning task* Π *and a natural number* $k \in \mathbb{N}$*,* loopless top-k planning *is the problem of determining a set of plans* $P \subseteq P_\Pi^{\ell\ell}$ *such that the following conditions hold:*

C1. there exists no plan $\pi' \in P_\Pi^{\ell\ell}$ *with* $\pi' \notin P$ *that is cheaper than some plan* $\pi \in P$*, and*

C2. $|P| = k$ *if* $|P_\Pi^{\ell\ell}| \geq k$*, and* $|P| = |P_\Pi^{\ell\ell}|$ *otherwise.*

Note that Definition 3 specifies ordinary top-k planning when $P_\Pi^{\ell\ell}$ is replaced with P_Π. Let P be the solution of a planner A for a given planning task Π. Then A is called *sound* for loopless top-k planning iff $P \subseteq P_\Pi^{\ell\ell}$ and P satisfies C1. A is called *complete* for loopless top-k planning iff A terminates and P satisfies C2.

[1] In order to expand all states S_g with a certain reachability cost g in the presence of zero-cost operators, symbolic planners like SYM-K first collect all states reachable with zero-cost operators. This reachability analysis is performed with a separate symbolic search (with state filtering) that computes all reachable states S_g from a given set S using all zero-cost operators (Torralba 2015).

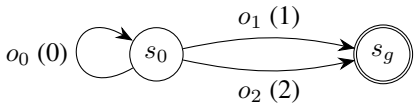

Figure 1: Counterexample for completeness of SYM-K-GNT with initial state s_0 and a unique goal state $s_g \in S_\star$.

Generate-and-Test Approach

A straightforward way to generate a set of desired plans is using a top-k planner with an anytime behavior like SYM-K (Speck, Mattmüller, and Nebel 2020), which enumerates all plans with increasing costs. The underlying idea of our generate-and-test approach, SYM-K-GNT, is to treat the ordinary top-k planner SYM-K as a black box by requesting a sufficiently large number of plans ($k = \infty$) and running it until a sufficient number of desired plans are found. In our case, we are interested in the k cheapest loopless plans. Therefore, for each newly reported plan π produced by SYM-K, we add π to our solution set P if it is loopless, and ignore it otherwise. Once k loopless plans are found, the algorithm terminates and returns P.

The generate-and-test approach SYM-K-GNT is sound since (a) it adds only loopless plans to the solution set P, and (b) it does not miss any cheaper plans, as SYM-K reports plans in non-decreasing order of costs.

SYM-K-GNT is a straightforward approach to loopless top-k planning, but it comes with some drawbacks: First, the performance of SYM-K-GNT strongly depends on the number of ignored plans, since the generation of these undesired plans can lead to a large overhead (see empirical evaluation). Second, and most critically, SYM-K can report infinitely many plans with the same cost, only finitely many of which are accepted by the looplessness test. This occurs if a plan contains a loop with zero-cost operators: Consider a simple planning task Π with three operators $\mathcal{O} = \{o_0, o_1, o_2\}$ with $\mathrm{cost}(o_i) = i$, where the induced transition system is shown in Figure 1. In this example, there are infinitely many optimal plans with a cost of 1, namely applying o_0 arbitrarily often before applying o_1. SYM-K will report only these optimal plans with cost 1, of which only $\pi = \langle o_1 \rangle$ is loopless, and will never generate the other loopless plan $\pi' = \langle o_2 \rangle$ with cost 2. Thus, if the desired number of loopless plans is $k = 2$ ($P^{\ell\ell}_\Pi = \{\pi, \pi'\}$), SYM-K-GNT discards an infinite number of loopy plans and never terminates.

Proposition 1. SYM-K-GNT *is sound but not complete for loopless top-k planning.* □

Symbolic Search Approach

The second approach that we propose for loopless top-k planning, SYM-K-LL, modifies the SYM-K algorithm. In particular, the following two modifications are required to obtain a sound and complete algorithm.

Plan Reconstruction. The plan reconstruction procedure of SYM-K must keep track of all states it has visited since these are already part of the goal path being reconstructed. Those previously visited states will no longer be considered as possible parent states during reconstruction. This change avoids generating plans with loops and ensures that plan reconstruction always terminates starting from a certain target state. Note that it is possible to represent and handle the previously visited states concisely using BDDs which is the underlying data structure of SYM-K.

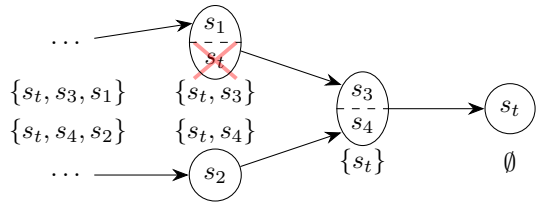

Figure 2: Example of the plan reconstruction of SYM-K-LL.

Example 1. *Figure 2 illustrates the reconstruction phase of* SYM-K-LL *starting from a state s_t. The reconstruction starts with $\{s_t\}$ and the empty set of visited states $\emptyset$. It computes the set of parents $\{s_3, s_4\}$ and updates the visited states to $\{s_t\}$. As states s_3 and s_4 have not been visited along this goal path, both are valid parent states. Thus, from both states a separate plan reconstruction is continued in a depth-first manner (greedy search). Let us consider s_3 next. The parents of s_3 are $\{s_1, s_t\}$, but s_t has already been visited along the current goal path, thus only s_1 is considered a valid parent state. This procedure is continued until k plans are found or no more paths can be reconstructed.*

Termination Criterion. The second modification concerns the termination criterion of SYM-K to ensure that SYM-K-LL terminates when no more loopless plans can be found. SYM-K terminates if either k plans are found or the open list contains only states that have already been expanded at least once and are not part of a goal path induced by a previously found plan (Speck, Mattmüller, and Nebel 2020). In other words, the second part of this termination criterion checks whether the states in the open list form a fixed point from where a goal path cannot be completed. This termination criterion also works for SYM-K-LL, because if k loopless plans have been found, it is safe to stop, and if no more plans can be found, neither can any more loopless plans. However, we need to strengthen the termination criterion so that SYM-K-LL stops when all existing loopless plans have been found, there are less than k of them, and all further goal paths contain loops.

The longest loopless plan can traverse each state of the finite state space $\mathcal{S}$ only once. Thus, for any loopless plan $|\mathcal{S}| - 1$ is an upper bound on the maximum plan length and $(|\mathcal{S}| - 1) \cdot \max_{o \in \mathcal{O}} \mathrm{cost}(o)$ is an upper bound on the maximum plan cost. Since we expand the states with increasing reachability costs in symbolic search, we can terminate as soon as SYM-K-LL exceeds the cost bound. This termination criterion could be strengthened by keeping track of the minimum path length to reach states in symbolic search and using the length bound directly. However, we consider both bounds to be interesting only for theoretical considerations, not for practical applications.

Theoretical Properties. Both introduced modifications together result in a sound and complete algorithm SYM-K-LL for loopless top-k planning, which avoids the generation of loopy plans and can handle zero-cost loops.

Theorem 1. SYM-K-LL *is sound and complete for loopless top-k planning.*

Proof. By construction of the plan reconstruction procedure, no returned plan can contain a loop. C1 of Definition 3 is satisfied because SYM-K-LL, like SYM-K, expands all reachable states with increasing cost and all possible goal paths are considered. This proves soundness.

The plan reconstruction of SYM-K-LL terminates because it retains the states already visited on each path and does not revisit these states. Thus, at some point, all possible states on each reconstructed goal path have been considered. SYM-K-LL terminates because the plan reconstruction calls terminate and at some point either k loopless plans have been found or the upper cost bound has been reached. Finally, C2 of Definition 3 is satisfied because SYM-K-LL, like SYM-K, reconstructs all possible plans, but ignores loopy plans. This proves completeness. □

Empirical Evaluation

We empirically evaluate the two proposed approaches SYM-K-LL and SYM-K-GNT for loopless top-k planning on a benchmark set consisting of 2262 planning tasks from 74 domains taken from the optimal track of the International Planning Competitions 1998-2018.[2] For each task, the time limit is set to 30 minutes and the memory limit is set to 4 GB. As a measure of planner performance, we use k-coverage: the number of tasks a planner was able to solve within the given bounds for a given number of requested loopless plans k. We set the number of requested loopless plans to $K = 10^4$ and derive the k-coverage values for all $k \leq K$, given the anytime behavior of the planners.

Figure 3 shows that, for $k > 1$ and each search direction, SYM-K-LL can solve substantially more instances than SYM-K-GNT. Regarding the search direction, we can see that for both approaches, bidirectional search dominates forward and backward search. The latter is not surprising since the same is also the case for ordinary symbolic planners (Edelkamp, Kissmann, and Torralba 2015; Torralba et al. 2017; Speck, Geißer, and Mattmüller 2020) and for the underlying symbolic top-k planner SYM-K.

A natural question is to what extent the gap between the k-coverage of SYM-K-GNT and SYM-K-LL is due to the fact that only SYM-K-LL is complete in the presence of zero-cost loops. Considering only domains without zero-cost operators, we find that the advantage of SYM-K-LL in terms of k-coverage is almost as large as in Figure 3. Thus, the disadvantage of SYM-K-GNT originates mainly from the unnecessary generation of loopy plans and not from the lack of completeness in some domains. Finally, it should be mentioned that SYM-K-GNT can be the better choice in some domains that have few or no plans with loops and do not require the more complex plan reconstruction of SYM-K-LL.

[2] Available online: https://github.com/speckdavid/symk

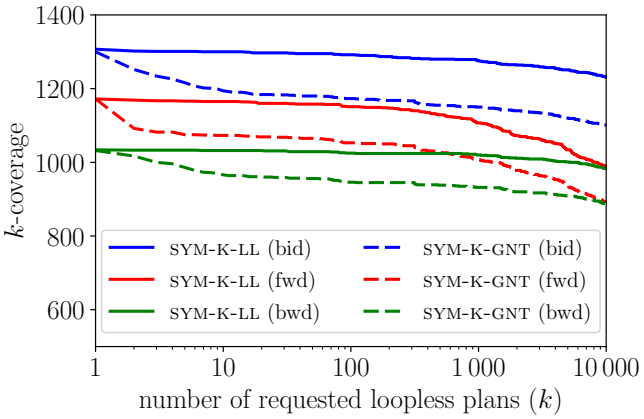

Figure 3: The overall k-coverage of SYM-K-LL and SYM-K-GNT with forward (fwd), backward (bwd) and bidirectional (bid) search.

Related Work

The field of diverse planning has the objective of generating a set of interesting and diverse plans according to a diversity measure (Coman and Muñoz-Avila 2011; Nguyen et al. 2012; Goldman and Kuter 2015; Katz and Sohrabi 2020). The underlying idea is to provide a good set of plans that offers reasonable alternatives. Katz, Sohrabi, and Udrea (2020), proposed to search for practically useful plan sets by defining an equivalence relation over plans to determine only one representative of each equivalence class. However, their work does not provide a general solution to find only unique representatives of the equivalence classes. In our case, each loopless plan forms the representative of a separate equivalent class and our proposed symbolic approach is the first complete approach that can generate loopless plans with increasing costs, allowing the search for more practically useful sets. Note that it is possible to further restrict the set of loopless plans by considering the plans generated by SYM-K-LL (or SYM-K-GNT) and filtering out the undesirable ones (e.g., plan permutations) or to optimize the set of loopless plans for a particular diversity metric.

Conclusion

In this paper, we introduced loopless top-k planning, where the solution set contains only plans that do not visit any state more than once. Such loopless plans are more useful in many real-world applications, as plans with loops are dominated by a loopless representative. Since there can be infinitely many optimal plans in the presence of zero-cost loops, ordinary top-k planners are of limited use, as they generate undesired loopy plans that can prevent more meaningful plans from being found. We have proposed two approaches to loopless top-k planning, a generate-and-test approach, SYM-K-GNT, which is sound but not complete, and a symbolic search approach, SYM-K-LL, which is sound and complete. The empirical evaluation shows that it is possible to determine a set of k best loopless plans in various domains using both approaches, with SYM-K-LL performing substantially better overall in terms of k-coverage.

Acknowledgments

This research was partially supported by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under grant agreement no. 952215. David Speck was supported by the German Research Foundation (DFG) as part of the EPSDAC project (MA 7790/1-1).

References

Aljazzar, H.; and Leue, S. 2011. K*: A Heuristic Search Algorithm for Finding the K Shortest Paths. *AIJ*, 175(18): 2129–2154.

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

Bock, F.; Kanter, H.; and Haynes, J. 1957. *An Algorithm (the r-th Best Path Algorithm) for Finding and Ranking Paths through a Network*. Armour Research Foundation.

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

Coman, A.; and Muñoz-Avila, H. 2011. Generating Diverse Plans Using Quantitative and Qualitative Plan Distance Metrics. In *Proc. AAAI 2011*, 946–951.

Edelkamp, S.; Kissmann, P.; and Torralba, Á. 2015. BDDs Strike Back (in AI Planning). In *Proc. AAAI 2015*, 4320–4321.

Eppstein, D. 1998. Finding the k Shortest Paths. *SICOMP*, 28(2): 652–673.

Gerevini, A. E.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners. *AIJ*, 173(5–6): 619–668.

Goldman, R. P.; and Kuter, U. 2015. Measuring Plan Diversity: Pathologies in Existing Approaches and A New Plan Distance Metric. In *Proc. AAAI 2015*, 3275–3282.

Hershberger, J.; Maxel, M.; and Suri, S. 2007. Finding the k Shortest Simple Paths: A New Algorithm and Its Implementation. *ACM Transactions on Algorithms*, 3(4): 45:1–45:19.

Höller, D. 2021. Translating Totally Ordered HTN Planning Problems to Classical Planning Problems Using Regular Approximation of Context-Free Languages. In *Proc. ICAPS 2021*, 159–167.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. In *Proc. AAAI 2020*, 9892–9899.

Katz, M.; Sohrabi, S.; and Udrea, O. 2020. Top-Quality Planning: Finding Practically Useful Sets of Best Plans. In *Proc. AAAI 2020*, 9900–9907.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *Proc. ICAPS 2018*, 132–140.

Kissmann, P.; Edelkamp, S.; and Hoffmann, J. 2014. Gamer and Dynamic-Gamer – Symbolic Search at IPC 2014. In *IPC-8 planner abstracts*, 77–84.

Koehler, J.; and Schuster, K. 2000. Elevator Control as a Planning Problem. In *Proc. AIPS 2000*, 331–338.

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

Nguyen, T. A.; Do, M. B.; Gerevini, A.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *AIJ*, 190: 1–31.

Riabov, A. V.; Sohrabi, S.; and Udrea, O. 2014. New Algorithms for The Top-K Planning Problem. In *ICAPS 2014 Scheduling and Planning Applications woRKshop*, 10–16.

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

Speck, D.; Geißer, F.; and Mattmüller, R. 2018. SYMPLE: Symbolic Planning based on EVMDDs. In *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 *Proc. ICAPS 2020*, 263–271.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In *Proc. AAAI 2020*, 9967–9974.

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 *IPC-8 planner abstracts*, 105–109.

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

Trüg, S.; Hoffmann, J.; and Nebel, B. 2004. Applying Automatic Planning Systems to Airport Ground-Traffic Control – A Feasibility Study. In *Proc. KI 2004*, 183–197.

Yen, J. Y. 1971. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11): 712–716.