Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

When Perfect Is Not Good Enough: On the Search Behaviour of Symbolic Heuristic Search

David Speck
University of Freiburg, Germany
speckd@cs.uni-freiburg.de

Florian Geißer
The Australian National University, Australia
florian.geisser@anu.edu.au

Robert Mattmüller
University of Freiburg, Germany
mattmuel@cs.uni-freiburg.de

Abstract

Symbolic search has proven to be a competitive approach to cost-optimal planning, as it compactly represents sets of states by symbolic data structures. While heuristics for symbolic search exist, symbolic bidirectional *blind* search empirically outperforms its heuristic counterpart and is therefore the dominant search strategy. This prompts the question of why heuristics do not seem to pay off in symbolic search. As a first step in answering this question, we investigate the search behaviour of symbolic heuristic search by means of BDDA*. Previous work identified the partitioning of state sets according to their heuristic values as the main bottleneck. We theoretically and empirically evaluate the search behaviour of BDDA* and reveal another fundamental problem: we prove that the use of a heuristic does not always improve the search performance of BDDA*. In general, even the perfect heuristic can exponentially deteriorate search performance.

Introduction

Over the past decade, both explicit search and symbolic search have proven to be strong and competitive approaches to cost-optimal planning. While explicit search is mainly based on variations of A* (Hart, Nilsson, and Raphael 1968) in combination with strong and efficient heuristics (Franco et al. 2018; Seipp, Keller, and Helmert 2020), the dominant search strategy of modern symbolic planners is bidirectional search without any heuristic (Torralba, Linares López, and Borrajo 2016; Speck, Geißer, and Mattmüller 2018b). In symbolic search, sets of states are represented by compact data structures, such as binary decision diagrams (BDDs) (Bryant 1986), which make it possible to perform an exhaustive search. Clearly, an important question is whether symbolic search can be further improved with heuristics like it is the case with explicit search. Interestingly, there exist a variety of generalizations of A* based on different symbolic data structures (Edelkamp and Reffel 1998; Hansen, Zhou, and Feng 2002; Speck, Geißer, and Mattmüller 2018a). In addition, multiple heuristics can be computed and represented with symbolic data structures, resulting in state-of-the-art heuristics for explicit heuristic search (Edelkamp 2002; Franco et al. 2017; Moraru et al. 2019). Thus, all the ingredients are present to allow a symbolic planner utilizing heuristics, as explicit planners do. However, empirical evaluations of symbolic heuristic search show that the use of heuristics in symbolic search can improve, but also impair the performance, depending on the domain (Torralba 2015; Torralba et al. 2017). In the literature, this is often explained by the effort required to perform additional symbolic (arithmetic) operations that divide the states according to the corresponding heuristic values (Jensen, Veloso, and Bryant 2008). While this is indeed a problem of symbolic heuristic search, it is still not clear why in general the benefits of pruning states based on heuristic values not outweigh the additional effort required to incorporate heuristics. This is the question we address in this paper. We expose another fundamental problem of symbolic heuristic search: good distance estimations are not the correct quantity to improve the search performance of symbolic heuristic search. More precisely, we show that heuristics, even the perfect heuristic, can increase the representation size of sets of states and thus impair the search performance of symbolic search.

Edelkamp and Reffel (1998) presented the first symbolic version of A*, called BDDA*. The underlying idea is to represent a heuristic function with multiple BDDs. For each heuristic value h, a separate BDD is used to represent the states S_h with heuristic value h. In BDDA*, all states S_g reachable with cost g are partitioned according to their heuristic value by computing the intersections of S_g and S_h for each heuristic value h. This was identified as a bottleneck because multiple arithmetic operations have to be performed during search (Jensen, Veloso, and Bryant 2008). Therefore, multiple extensions to BDDA* have been published such as Lazy BDDA* (Torralba 2015), which delays the heuristic evaluation as long as possible, or SetA* (Jensen, Veloso, and Bryant 2008), which encodes the heuristic values as preconditions of actions resulting in multiple actions with costs according to the heuristic values. While this saves expensive arithmetic operations during search, it may blow up the action space. Empirical evaluations show that all versions of BDDA* perform better than blind search in some domains and worse in other domains. Overall, the dominant search strategy for symbolic planning remains bidirectional search without any heuristic (Torralba, Linares López, and Borrajo 2016; Speck, Geißer, and Mattmüller 2018a).

We focus on symbolic heuristics based on BDDs, more precisely on BDDA* and show that pruning states based

on heuristic values can deteriorate the search performance when representing sets of states as decision diagrams. Most of our results generalize to the most prominent variations of BDDA*, as well as symbolic A* based on other types of decision diagrams. In contrast to explicit A*, where every consistent heuristic can only reduce the number of necessary node expansions (up to tie breaking) and thus the search effort compared to blind search, we show that in BDDA* no such guarantee exists. More precisely:
In symbolic search, the search effort is not directly related to the number of explicit states that have to be expanded.

Rather, the size of expanded BDDs, i.e., number of BDD nodes, representing expanded states determines the search effort and thus the runtime. Torralba (2015) showed empirically that the correlation between BDD nodes and represented states is less significant than the correlation of BDD nodes and runtime. In this paper, we prove that in theory even under the best possible and unrealistic circumstances, namely the perfect heuristic, the search effort of BDDA* can be exponentially larger than the search effort of symbolic search without heuristic, and vice versa. This shows that heuristics in symbolic search can exponentially increase or decrease search performance. In particular, it is not the case that the use of a heuristic always improves search performance of symbolic search. Our empirical evaluation is consistent with these theoretical results. Understanding symbolic heuristic search is an important step in understanding why symbolic bidirectional search without heuristics is the dominant symbolic search strategy in planning. Finally, we discuss the implications of our theoretical and empirical findings and possible directions to improve BDDA*. Our theoretical and practical results show why recent advances in *explicit* bidirectional heuristic search cannot be applied directly to *symbolic* bidirectional heuristic search (Holte et al. 2016; Chen et al. 2017).

Preliminaries

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

Definition 1 (Classical Planning Task). A *classical planning task* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$ consisting of the following four components: $\mathcal{V}$ 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 $d \in D_v$. If variable v is a binary variable ($D_v = \{0, 1\}$), we refer with $\neg v$ to $(v, 0)$ and with v to $(v, 1)$. A partial variable assignment over $\mathcal{V}$ is a consistent set of facts. If s assigns a value to each $v \in \mathcal{V}$, s is called a state. States and partial variable assignments are functions which map variables to values, i.e., $s(v)$ is the value of variable v in state s (analogous for partial variable assignments). $\mathcal{O}$ is a set of operators, where an operator is a pair $o = \langle pre_o, \mathit{eff}_o \rangle$ of partial variable assignments called preconditions and effects, respectively. The state $\mathcal{I}$ is called the initial state and the partial state $\mathcal{G}$ specifies the goal condition, which defines all possible goal states $S_\star$. With $\mathcal{S}$ we refer to the set of all possible states defined over $\mathcal{V}$, and with $|\Pi|$ we refer to the size of planning task Π, i.e., the number of operators and facts.

We call an operator $o \in \mathcal{O}$ applicable in state s iff pre_o is satisfied in s, i.e., $s \models pre_o$. Applying operator o in state s results in a state s' where $s'(v) = \mathit{eff}_o(v)$ for all variables $v \in \mathcal{V}$ for which eff_o is defined and $s'(v) = s(v)$ for all other variables. We also write $s[o]$ for s'. The objective of classical planning is to determine a plan which is defined as follows.

Definition 2 (Plan). A *plan* $\pi = \langle o_0, \dots, o_{n-1} \rangle$ for planning task Π is a sequence of applicable operators which generates a sequence of states $s_0, \dots, s_n$, where $s_0 = \mathcal{I}$, $s_n \in S_\star$ is a goal state and $s_{i+1} = s_i[o_i]$ for all $i = 0, \dots, n-1$. Such a plan is considered to be optimal if there is no shorter plan.[1]

A set of states $S \subseteq \mathcal{S}$ can be represented by its characteristic function $\chi_S : \mathcal{S} \mapsto \{0, 1\}$. More precisely, the characteristic function χ_S *characterizes* the set of states S by determining whether a state belongs to S or not, mapping each state $s \in S$ to true and all other states to false, i.e., $\chi_S(s) = 1$ if $s \in S$ and $\chi_S(s) = 0$ otherwise ($s \notin S$). Similarly, operators can be represented as so-called *transition relations (TRs)* which are sets of state pairs, namely predecessor and successor states. The characteristic function of a transition relation T representing a set of operators $O \subseteq \mathcal{O}$ is a function $\chi_T : \mathcal{V} \times \mathcal{V}' \mapsto \{0, 1\}$ which maps all pairs of states (s, s') to true iff successor s' is reachable from predecessor s by applying an operator $o \in O$. Given a set of states S and a TR T, the *image* (*preimage*) operator computes the set of successor (predecessor) states S' of S through T. The most common data structure for the representation of characteristic functions and the efficient application of (pre-)image operations are (reduced and ordered) binary decision diagrams (Bryant 1986).

Definition 3 (Binary Decision Diagram). A *binary decision diagram (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[2] variable $v \in \mathcal{V}$ and has two successors, 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. We denote with $|B|$ the *size* of BDD B, i.e., the number of inner nodes of B.

A BDD is called *ordered* if on all paths from the root to a sink variables appear in the same order. A BDD is called *reduced* if isomorphic subgraphs are merged and any node is eliminated whose two children are isomorphic. For fixed variable orders, reduced and ordered BDDs are unique. From now on we only talk about reduced and ordered BDDs and assume a fixed variable order.

Symbolic Search

Symbolic search is a state space exploration technique that uses efficient data structures to represent and manipulate sets of states (McMillan 1993). Like explicit A*, BDDA*

[1] We consider unit costs without loss of generality.

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

(Edelkamp and Reffel 1998) expands states $s \in \mathcal{S}$ in ascending order of their *f-values* $f(s) = g(s) + h(s)$, where the *g-value* $g(s)$ denotes the cost of reaching state s and the *h-value* $h(s)$ denotes a heuristic estimate of state s. In contrast to explicit A*, in BDDA*, sets of states with the same g and h-value are represented as one BDD and expanded at once.[3] Heuristic functions h are precomputed and represented as multiple BDDs H_i, one per heuristic value i. A heuristic is a function $h : \mathcal{S} \mapsto \mathbb{N} \cup \{\infty\}$, which depending on the search direction estimates the cost to reach a goal state (forward search) or the initial state (backward search) from a state $s \in \mathcal{S}$. The perfect heuristic $h^\star$ maps each state s to the cost of the cheapest path from s to any goal state (the initial state) and the blind heuristic h^0 maps each state s to 0. Heuristic h is called admissible if h never overestimates the cost of reaching a goal state (the initial state), i.e., $h(s) \leq h^\star(s)$ for all $s \in \mathcal{S}$. Heuristic h is called consistent if $h(s) \leq h(s[o]) + 1$ for all $s \in \mathcal{S}$ and $o \in \mathcal{O}$ where $s[o]$ is defined, and $h(s) = 0$ for all $s \in S_\star$ (for $s = \mathcal{I}$). Note that every consistent heuristic is also admissible (Pearl 1984). Heuristics h^0 and $h^\star$ are two examples of consistent heuristics. Explicit A* and BDDA* are guaranteed to find optimal solutions when using a consistent heuristic. As usual, we assume that BDDA* is equipped with a consistent heuristic and uses a tie-breaking rule in favor of states with smaller g-values (Kissmann and Edelkamp 2011; Torralba 2015). The tie-breaking rule ensures a deterministic expansion order and avoids the expansion of two BDDs with the same g and h-values. Thus, in each expansion step, the corresponding set of states represented as BDD can be uniquely described by the corresponding g and h-values.

Forward BDDA* is carried out from the initial state towards a goal state and *backward* BDDA* is carried out from the goal states towards the initial state. In each expansion step, the BDD with minimum f-value (preferably lower g-values) is expanded, resulting in a BDD B that represents all successor (predecessor) states. First, all states already expanded (stored in the corresponding closed list) are removed from B before the BDD B is partitioned based on the BDDs H_i representing the heuristic values. Finally, BDDA* terminates once a BDD is found whose intersection with the goal (initial state) is not empty (Kissmann and Edelkamp 2011; Torralba 2015). In *bidirectional* BDDA*, both forward and backward BDDA* are performed simultaneously, thus maintaining two BDDA* searches with separate open and closed lists. 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 direction is expanded which is already contained in the closed list of the search in the opposite direction, a goal path is found. In general, all strategies which switch iteratively between both search directions guarantee optimality if the termination criterion is chosen accordingly (Pohl 1969). To simplify terminology, from here on we refer with BDDA* to *forward* BDDA*.

[3] In practice, sometimes more than one BDD is used to represent all states with the same g and h-values (Torralba et al. 2014).

Theoretical Results

This section is structured as follows. First, we introduce the concept of *expansion size* of BDDA*, which determines the search effort. Second, we show that a heuristic, even the perfect heuristic, can improve or impair the search performance of *unidirectional* BDDA*. Finally, we generalize those results to the case of *bidirectional* BDDA*.

Search Performance of BDDA*

The performance of explicit heuristic search, such as A*, is usually measured by the number of expanded search nodes (Helmert and Röger 2008). In general, each of these search nodes represents a single explicit state. In contrast, in symbolic search whole sets of states are expanded at once by performing BDD operations. The runtime of such operations depends on the size of the involved BDDs (Kissmann and Edelkamp 2011). Since the image and preimage operations are usually the most time-consuming process in symbolic search (Torralba 2015), we estimate the search effort in terms of expanded BDDs rather than generated BDDs. This is why we can approximate the performance of BDDA* by the cumulative sizes of the BDDs that are expanded during search. In the bidirectional case this approximation is based on the cumulative sizes of the BDDs which are expanded by both search directions. Similar to Helmert and Röger (2008), we are interested in lower bounds and estimate the search effort of BDDA* conservatively based on the cumulative size of BDDs, which must always be expanded by BDDA* before an optimal solution is found. Since BDDA* breaks ties in favor of states with smaller g-values, the number of expanded states represented as BDDs is larger than strictly necessary. However, this does not affect the results of this paper.

Definition 4 (BDDA* Expansion Size). We denote with $B_{\Pi,h}(i,j)$ the unique BDD that is expanded by BDDA* with g-value i and h-value j for a planning task $\Pi = \langle \mathcal{V}, \mathcal{I}, \mathcal{O}, \mathcal{G} \rangle$ and heuristic function h. With $size_{\Pi,h}(i,j)$ we refer to the size of this BDD, i.e., $size_{\Pi,h}(i,j) = |B_{\Pi,h}(i,j)|$. Finally, with $\mathit{effort}(\Pi, h)$ we refer to the *expansion size* of BDDA*, i.e., the sum of the sizes of all BDDs $B_{\Pi,h}(i,j)$ with the property $i + j \leq h^\star(\mathcal{I})$ and $i < h^\star(\mathcal{I})$.

In particular, it is possible that BDD $B_{S'}$ can be exponentially larger than BDD B_S although the set of states S' is a strict subset of S, i.e., $S' \subsetneq S$. This is one of the reasons why theoretical results from explicit A* cannot be directly transferred to its symbolic counterparts.

The Unidirectional Case

It is known that, up to tie breaking, A* never expands more search nodes using any consistent heuristic instead of the blind heuristic h^0 (Pearl 1984). A consistent heuristic never impairs the search performance of explicit A* in terms of so-called "must-expand" nodes n with the property $f(n) < h^\star(\mathcal{I})$. In BDDA*, determining the heuristic values of states by partitioning the corresponding BDDs was identified as a main problem that reduces search performance (Jensen, Veloso, and Bryant 2008). While this is indeed a bottleneck of BDDA*, we show in the following another fundamental

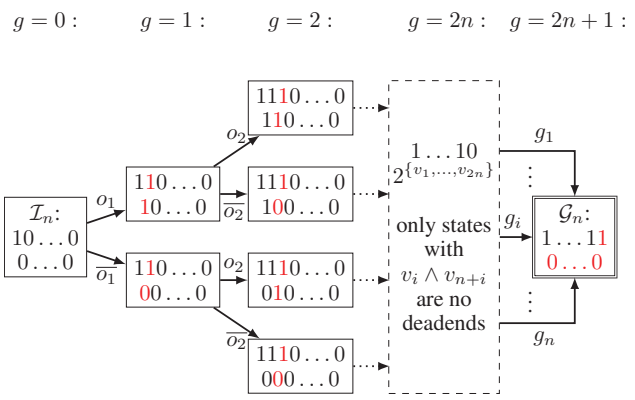

Figure 1: Visualization of the induced transition system of the family of planning tasks Π_n. Nodes represent states where the top row denotes the values of the x variables in ascending order and the bottom row denotes the values of the v variables in ascending order.

problem: we prove that even when equipped with the perfect heuristic $h^\star$ the expansion size of BDDA* can be exponentially larger than the expansion size of BDDA* equipped with the blind heuristic h^0. In order to prove this statement we introduce the following family of planning tasks Π_n.

Definition 5 (Planning Task Family Π_n). $\Pi_n = \langle \mathcal{V}_n, \mathcal{I}_n, \mathcal{O}_n, \mathcal{G}_n \rangle$ is a family of planning tasks with binary variables and the following components.

- $\mathcal{V}_n = \{v_i \mid 1 \leq i \leq 2n\} \cup \{x_i \mid 0 \leq i \leq 2n+1\}$
- $\mathcal{O}_n = \{o_i, \overline{o_i} \mid 1 \leq i \leq 2n\} \cup \{g_i \mid 1 \leq i \leq n\}$
  - $o_i = \langle x_{i-1} \wedge \neg x_i, x_i \wedge v_i \rangle$
  - $\overline{o_i} = \langle x_{i-1} \wedge \neg x_i, x_i \wedge \neg v_i \rangle$
  - $g_i = \langle x_{2n} \wedge v_i \wedge v_{n+i}, x_{2n+1} \wedge \bigwedge_{1 \leq j \leq 2n} \neg v_j \rangle$
- $\mathcal{I}_n = \{x_0\}$
- $\mathcal{G}_n = x_{2n+1}$

Figure 1 depicts the induced transition system of Π_n, which shows that the only reachable goal state[4] is reachable with minimal cost of $2n+1$. Note that the size of Π_n is linear in n. Next, we prove that BDDA* with h^0 expands the same amount of BDDs as BDDA* with $h^\star$.

Lemma 1. *Given a planning task Π_n, BDDA* with h^0 expands the same number of BDDs as BDDA* with $h^\star$.*

Proof. The only reachable goal state is reachable with cost $2n+1$. Thus, BDDA* with h^0 expands a total of $2n+1$ BDDs, i.e., $B_{\Pi_n,h^0}(0,0), B_{\Pi_n,h^0}(1,0), \ldots, B_{\Pi_n,h^0}(2n,0)$. BDDA* with $h^\star$ expands the BDDs $B_{\Pi_n,h^\star}(0,2n+1), B_{\Pi_n,h^\star}(1,2n), \ldots, B_{\Pi_n,h^\star}(2n,1)$, which are again a total of $2n+1$ BDDs. □

While Lemma 1 shows that the numbers of expanded BDDs using $h^\star$ and h^0 solving Π_n are equal, we now prove

[4]Due to the single reachable goal state, all theoretical results are independent of whether the check for goal states is performed when states are expanded or generated.

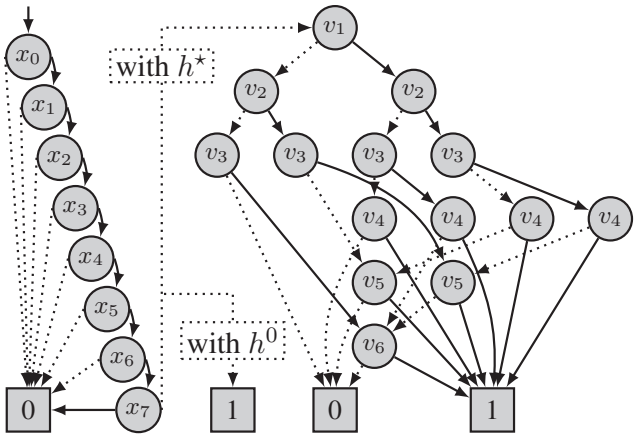

Figure 2: BDDs $B_{\Pi_3,h^0}(6,0)$ and $B_{\Pi_3,h^\star}(6,1)$ which are expanded last by BDDA* solving planning task Π_3.

that in every expansion, the size of the expanded BDD using h^0 is less than or equal to the size of the expanded BDD using $h^\star$. Lemmas 1 and 2 show that using a heuristic does not necessarily improve search performance, even under the best possible circumstances, the perfect heuristic $h^\star$.

Lemma 2. *Given planning task Π_n, in each expansion, the BDD expanded using h^0 is at most as large as the BDD expanded using $h^\star$, i.e., $size_{\Pi_n,h^0}(i,0) \leq size_{\Pi_n,h^\star}(i, 2n+1-i)$ for all $0 \leq i \leq 2n$.*

Proof. The BDD $B_{\Pi_n,h^0}(0,0)$, which represents the initial state, has size $size_{\Pi_n,h^0}(0,0) = |\mathcal{V}_n|$, because the initial state is a fully specified state and exactly one node is required for each variable to determine whether this variable is true or false. BDD $B_{\Pi_n,h^0}(i,0)$ has size $size_{\Pi_n,h^0}(i,0) = |\mathcal{V}_n| - i$, since in each step, the nodes determining the values of variables $v_{\leq i}$ are no longer necessary. With $h^\star$, however, in every step, all states are "removed" which do not lead to a goal state. Thus, the corresponding BDD $B_{\Pi_n,h^\star}(i, 2n+1-i)$ either represents the same set of states as $B_{\Pi_n,h^0}(i,0)$ or a subset of states. However, every non-empty subset requires additional nodes branching over variables v_i which can only increase the size of the BDD. □

Lemmas 1 and 2 are independent of the variable order and show that, in general, the search performance can deteriorate when using $h^\star$ instead of h^0. Finally, Theorem 3 shows that the search performance of BDDA* for a given variable order can be exponentially *worse* if $h^\star$ is used over h^0. It is extremely difficult to determine a (static) variable order that leads to small BDDs a priori. In practice, it is "almost impossible" to predict the size of BDDs before creating them (Kissmann and Hoffmann 2014). Furthermore, there are functions which require exponentially many nodes to represent independent of the variable ordering (Bryant 1986; Edelkamp and Kissmann 2011).

Theorem 3. *Using $h^\star$ instead of h^0 can increase the expansion size of BDDA* exponentially in the size of the planning task $|\Pi|$ for a given variable ordering.*

Proof. Consider the family of planning tasks Π_n and the variable ordering $x_0 \succ \cdots \succ x_{2n+1} \succ v_1 \succ \cdots \succ v_{2n}$. BDDA* with h^0 and BDDA* with $h^\star$ expand $2n+1$ BDDs (Lemma 1). In addition, in each step, the BDD expanded with h^0 has a linear size in n and is at most as large as the BDD expanded with $h^\star$ in the same step (Lemma 2). Thus, it is sufficient to show that the BDD expanded in the last step by BDDA* with $h^\star$ is exponentially larger than the BDD expanded by BDDA* with h^0. Considering BDDA* with h^0, the size of BDD $B_{\Pi_n,h^0}(2n,0)$ is $size_{\Pi_n,h^0}(2n,0) = |\mathcal{V}_n| - 2n$ as depicted in Figure 2 for $n = 3$ and explained in the proof of Lemma 2. Considering BDDA* with $h^\star$, we observe that only states where at least one pair v_i and v_{n+i} is true can lead to a goal state. Thus, the function $f := (v_1 \wedge v_{n+1}) \vee \cdots \vee (v_n \wedge v_{2n})$ has to be represented by $B_{\Pi_n,h^\star}(2n,1)$. Under the given variable order, representing f as a BDD requires exponentially many nodes (Kissmann 2012) in n, as depicted in Figure 2 for $n = 3$. □

Theorem 3 also holds for other versions of BDDA*, such as Lazy BDDA* and SetA*, because the expanded sets of states are exactly the same as for BDDA*. The delayed evaluation of Lazy BDDA* has no impact because at any step all states of the open list have the same g-values. SetA* only differs from BDDA* in that the state partitioning according to the heuristic values is encoded in the preconditions, but the expanded sets of states are exactly the same. Furthermore, Theorem 3 holds also for symbolic A* based on other decision diagrams. The functioning of ADDA* (Hansen, Zhou, and Feng 2002) and EVMDDA* (Speck, Geißer, and Mattmüller 2018a) is the same as BDDA* for planning tasks with unit costs and all decision diagrams have the same size when representing the relevant sets of states as characteristic functions.

Finally, for *unidirectional* BDDA*, it can be shown that a heuristic can also improve the search performance. In particular, we prove that the search effort of BDDA* can also be exponentially improved using $h^\star$ instead of h^0.

Theorem 4. *Using $h^\star$ instead of h^0 can decrease the expansion size of BDDA* exponentially in the size of the planning task $|\Pi|$ for a given variable ordering.*

Proof. Consider a family of planning tasks Π_n with exactly one plan π with a cost of $2n+2$. Furthermore, the size of the BDD, which represents the set of states S_{2n+1} reachable with a cost of $2n+1$, is exponential in the number of variables. It is possible to construct such a family of planning tasks with the same idea we have used for Π_n by 1) adding new variables y_i, $0 \leq i \leq 2n+2$, of which initially only y_0 is true, 2) adding operators $\omega_i = \langle y_{i-1} \wedge \neg y_i, y_i \rangle$, $1 \leq i \leq 2n+2$, forming a new path of length $2n+2$ from the initial state to a new goal state described by the new goal condition $\mathcal{G}_n = y_{2n+2}$, and 3) modifying the operators g_i to have the single effect x_{2n+1}. BDDA* with $h^\star$ expands, in each step, a BDD representing the explicit states along the sequence induced by the new operators ω_i. The size of BDDs representing explicit states is linear in the number of variables. BDDA* with h^0 expands a BDD representing the set of states S_{2n+1} in step $2n+1$, which is exponential in the number of variables given a certain variable ordering. □

Theorems 3 and 4 prove that even under the best possible and unrealistic circumstances the search effort of BDDA* can be exponentially larger than the search effort of symbolic blind search and vice versa. Most importantly, it is not the case that the use of a heuristic always improves the search performance of BDDA*. Heuristics can, to the same extent, harm and improve the search performance of BDDA*. In general, the presented theoretical results also generalize to the case where multiple BDDs are used to represent the set of states with the same g and h-values (Torralba et al. 2014), because maintaining a single BDD for every pair of g and h-values is a special case of this. Finally, it is worth mentioning that also the BDDs which represent the heuristic function can be exponential in size.

The Bidirectional Case

In general, there are multiple selection strategies to determine which search front should be expanded next in BDDA*. Our goal is to show that for different reasonable and practical selection strategies, our theoretical results of *unidirectional* BDDA* generalize to *bidirectional* BDDA* as well. We consider two selection strategies: *random*, which randomly selects a search direction, and *alternating*, which begins with the the forward search direction and then alternates between both search directions.

Theorem 5. *Using $h^\star$ instead of h^0 can increase or decrease the expansion size of* bidirectional *BDDA*, with the random selection strategy, exponentially in the size of the planning task $|\Pi|$ for a given variable ordering.*

Proof. A possible behaviour of the random selection strategy is to always select forward expansions, which results in *forward* BDDA*. Thus, the results of Theorems 3 and 4 hold also in this case. □

Theorem 6. *Using $h^\star$ instead of h^0 can increase or decrease the expansion size of* bidirectional *BDDA*, with the alternating selection strategy, exponentially in the size of the planning task $|\Pi|$ for a given variable ordering.*

Proof Sketch. The idea is to slightly modify the family of planning tasks we considered in the proofs of Theorems 3 and 4. It is possible to "append" a sequence of explicit states to the original goal state, which is no longer a goal state, leading to a new single goal state, so that each backward expanded BDD represents a single state and both search directions meet exactly at the old goal state. Bidirectional BDDA* with the alternating selection strategy expands in backward direction only BDDs representing single states independently of the heuristic and the forward direction has the search effort that we proved in Theorems 3 and 4. □

In practice, there exist multiple selection strategies with the goal of choosing the most promising search front next. SYMBA* (Torralba et al. 2014), a state-of-the-art symbolic planner, for example, tries to estimate the search time and BDD sizes of the next step in order to predict the search effort of both search directions.

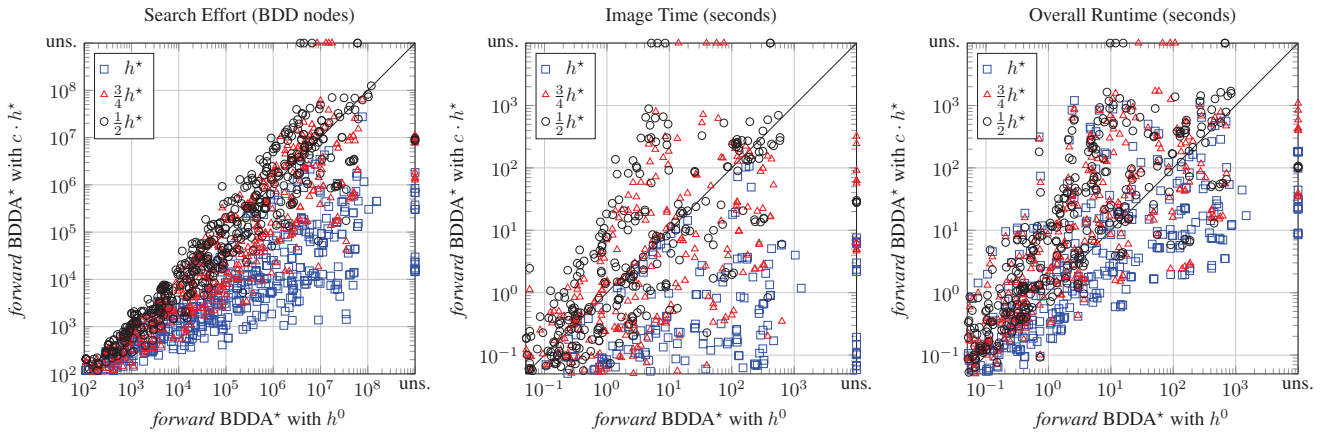

Figure 3: A comparison of *forward* BDDA* with the blind heuristic h^0 and with a given fraction perfect heuristics ($c \in \{1, \frac{3}{4}, \frac{1}{2}\}$) which includes the search effort (left), the cumulative time of the image operation (center) and the overall search time (right).

Empirical Results

We have seen theoretical results about the search behaviour of unidirectional and bidirectional BDDA*. In this section, we investigate whether we can observe this search behaviour in practice as well. To answer this question, we conducted experiments with the SYMBA* planner (Torralba et al. 2014; 2017), which is based on the FAST DOWNWARD planning system (Helmert 2006). We compare BDDA* with the blind heuristic h^0 with BDDA* with fraction perfect heuristics. A heuristic $c \cdot h^\star$ is called *fraction perfect* if it assigns to all states the values of the perfect heuristic multiplied by a constant $0 \leq c \leq 1$. Note that $0h^\star = h^0$ and $1h^\star = h^\star$ are important special cases. The benchmark set on which we evaluate BDDA* consists of all domains that do not contain axioms, conditional effects or zero-cost actions from the optimal track from the International Planning Competitions between 1998 and 2018.[5] In addition, we only report instances where it was possible to precompute the perfect heuristic. The perfect heuristic was computed with symbolic blind search with a time limit of 30 minutes and a memory limit of 4 GB. The closed list of unidirectional symbolic blind search represents the perfect heuristic for the opposite search direction. Overall, we compared forward and bidirectional BDDA* with the blind heuristic against BDDA* with different fraction perfect heuristics. Analogous to our theoretical results, all states with the same g and h-values are represented by a single BDD. Furthermore, we disabled mutex pruning (Alcázar and Torralba 2015) and used the default variable order of SYMBA* which is based on the work of Kissmann and Edelkamp (2011). We examine the search effort of BDDA*, i.e., the cumulative size of the BDDs representing the states that BDDA* must expand to determine an optimal plan. The cumulative image time, i.e., the total time required for the image and preimage operation, indicates that the search effort is an adequate quantity that affects the search performance of BDDA*. Finally, the overall runtime is taken into account to compare the actual performance of BDDA* with the different heuristics. Regardless of the experiment we set a time limit of 30 minutes and a memory limit of 4 GB and ignore the resources to calculate the perfect heuristic.

[5] In general, these concepts can be efficiently supported by symbolic planners (Kissmann, Edelkamp, and Hoffmann 2014; Speck et al. 2019), but they can also have a significant impact on the search behaviour.

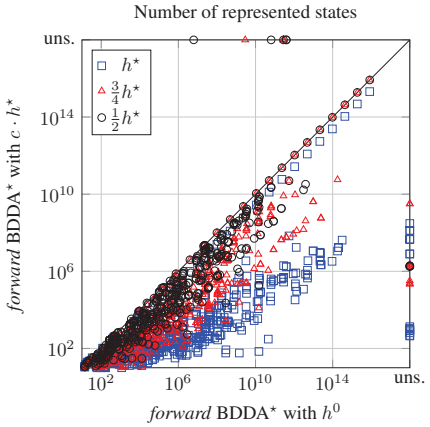

Figure 4: A comparison of the represented ("must-expand") states of *forward* BDDA* with the blind heuristic h^0 and with a given fraction perfect heuristics ($c \in \{1, \frac{3}{4}, \frac{1}{2}\}$).

The Unidirectional Case

Figure 3 (left) compares the search effort of *forward* BDDA* with the blind heuristic and the search effort of *forward* BDDA* with fraction perfect heuristics. In practice, the perfect heuristic almost always reduces the search effort. In some cases, however, there is no improvement or slight deterioration. It's hard to find an explanation why in practice the optimal heuristic helps to almost always reduce the search effort without analysing each case individually. A possible explanation is that there are only a few optimal plans in

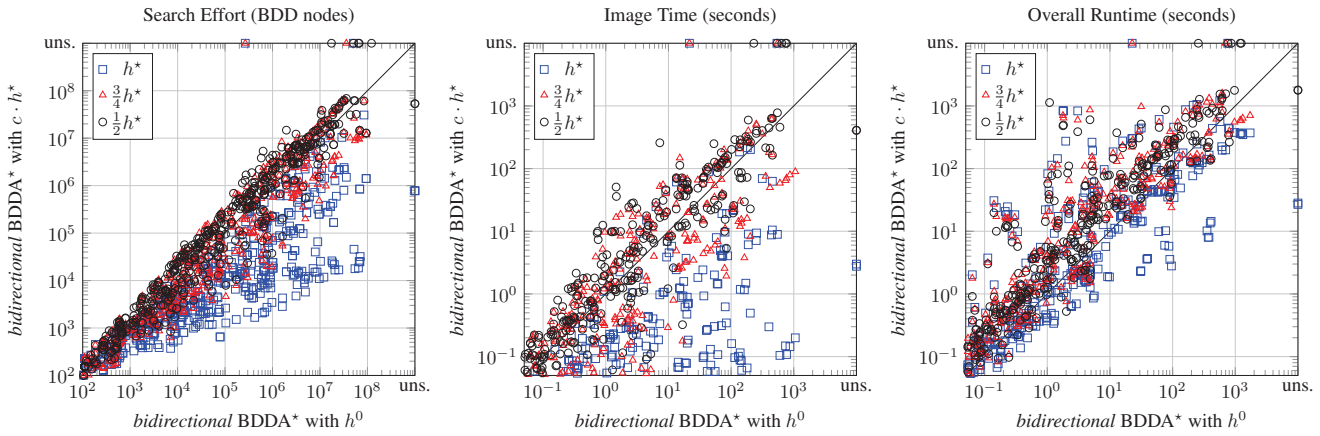

Figure 5: A comparison of *bidirectional* BDDA* with the blind heuristic h^0 and a given fraction perfect heuristics ($c \in \{1, \frac{3}{4}, \frac{1}{2}\}$) which includes the search effort (left), the cumulative time of the image operation (center) and the overall search time (right).

some domains; and the BDDs, which represent those explicit states on the induced state sequences, are compact. We were able to confirm this explanation only for a couple of domains (e.g. *airport* or *blocks*) using the top-k planner by Speck, Mattmüller, and Nebel (2020), which calculates the best k plans. Nevertheless, a perfect heuristic is an unrealistic assumption that is not achievable in practice. If we consider fraction perfect heuristics, namely $\frac{3}{4}h^\star$ and $\frac{1}{2}h^\star$, we can observe that the search effort of *forward* BDDA* can improve or deteriorate to the same extent. These empirical results are consistent with the presented theoretical results. Figure 4 shows that the number of "must-expand" states represented by the BDDs in BDDA* is always less than or equal when using fractional perfect heuristics, and that explicit A* would benefit from using such heuristics.

Figure 3 (center) shows the expansion time of BDDA*. More precisely, the cumulative time it takes to generate all successors with the image operation. It is possible to observe a correlation between the search effort and the expansion time, which also empirically shows that the search effort is an adequate quantity that influences the performance of BDDA*.

Figure 3 (right) shows the overall runtime of BDDA* without the time to compute the corresponding heuristic. If we compare the expansion time with the overall running time, we notice that BDDA* with fraction perfect heuristics has a higher time increase than BDDA* with h^0. Clearly, this can be traced back to the partitioning of sets of states according to heuristics values (Jensen, Veloso, and Bryant 2008), which can be time-consuming, since this procedure includes multiple conjunctions after each expansion.

Finally, Table 1 compares the search of BDDA* with and without heuristics per domain. We only consider instances which were solved by all configurations. An entry shows by which factor the corresponding heuristic has improved (< 1) or impaired (> 1) the search effort of BDDA* in that domain. This comparison shows that it appears to be domain dependent whether a heuristic helps BDDA*. This points

Algorithm	fwd. BDDA*		bid. BDDA*	
Domain (#Tasks) / Heuristic	$h^\star$	$\frac{1}{2}h^\star$	$h^\star$	$\frac{1}{2}h^\star$
AGRICOLA (1)	0.005	0.322	0.007	0.314
AIRPORT (20)	0.593	0.636	**1.042**	**1.110**
BARMAN (5)	0.001	**1.289**	0.001	**1.192**
BLOCKS (18)	0.001	0.062	0.004	0.039
DEPOT (2)	0.037	0.435	0.057	0.579
DRIVERLOG (8)	0.009	**1.152**	0.061	**4.894**
FLOORTILE (16)	0.000	0.045	0.010	**1.069**
FREECELL (14)	0.008	0.691	0.018	0.254
GRID (1)	0.058	0.288	**1.237**	**2.221**
GRIPPER (20)	0.354	**1.375**	0.794	**1.239**
HIKING (13)	0.012	0.631	0.055	**2.096**
LOGISTICS (18)	0.028	**1.261**	0.086	**2.950**
MICONIC (72)	0.098	**4.060**	0.399	**2.838**
MOVIE (30)	**1.082**	**1.864**	**2.064**	**1.409**
MPRIME (4)	0.051	0.186	0.400	**1.031**
MYSTERY (5)	0.118	0.289	0.535	0.923
NOMYSTERY (11)	0.000	0.005	0.002	0.128
OPENSTACKS (10)	0.081	**4.279**	0.082	**2.984**
ORGANIC (17)	0.870	0.870	0.585	0.585
PATHWAYS (4)	0.080	0.809	0.072	0.703
PIPESWORLD (14)	0.014	0.257	0.114	0.396
PSR-SMALL (50)	0.961	**1.204**	**1.081**	**1.184**
ROVERS (10)	0.018	**5.180**	0.046	**4.163**
SATELLITE (7)	0.004	0.511	0.045	**2.048**
SCANALYZER (21)	0.002	**1.225**	0.008	**2.104**
STORAGE (12)	0.006	0.119	0.018	0.301
TERMES (1)	0.011	0.513	0.098	**2.568**
TETRIS (2)	0.036	0.247	0.116	0.283
TIDYBOT (5)	0.008	0.193	0.337	0.462
TPP (8)	0.075	**5.414**	0.070	**4.374**
TRANSPORT (10)	0.014	0.432	0.048	**1.888**
TRUCKS (9)	0.001	0.178	0.001	0.156
VISITALL (14)	0.033	**1.386**	0.128	**2.479**
WOODWORKING (25)	0.009	**2.166**	0.026	0.395
ZENOTRAVEL (7)	0.018	0.392	0.155	**2.191**
GEOM. MEAN	0.134	**1.142**	0.280	**1.530**

Table 1: Per domain comparison: an entry shows by which factor the corresponding heuristic has improved (< 1) or impaired (> 1) the search effort of BDDA* in that domain.

to the fact that the structure of the reachable search space plays a central role if a heuristic helps or impairs BDDA*. However, as Kissmann and Hoffmann (2014) already investigated, it appears to be "almost impossible" to predict the size of the relevant BDDs a priori.

The Bidirectional Case

In bidirectional search, SYMBA*'s selection strategy predicts whether a forward search step or a backward search step takes less time and results in smaller BDDs. We have used this selection strategy for our final empirical evaluation to be as close as possible to the state of the art of symbolic bidirectional search. Experiments with an alternating selection strategy showed a similar picture with regard to the comparison between blind and heuristic search. Figure 5 (left) shows the search effort of BDDA* with and without heuristics. The results of bidirectional BDDA* are similar to the results of unidirectional BDDA* (Figure 3). However, in the bidirectional case the perfect heuristic helps less, especially with regard to the number of tasks that BDDA* could only solve with the perfect heuristic.

The expansion time (center) shown in Figure 5 empirically shows that the sizes of the BDDs also have an important influence on the runtime for bidirectional BDDA*.

Figure 5 (right) shows the total runtime of bidirectional BDDA* ignoring the time to compute the corresponding heuristic. We can observe that all heuristics, even the perfect heuristic, can help or harm in some instances.

The right column of Table 1 shows that the effect of a heuristic in bidirectional BDDA* depends on the domain. Looking at the geometric mean, we see that fraction perfect heuristics perform worse in bidirectional search than in unidirectional search. This can be traced back to the explicit case in which many bidirectional heuristic search algorithms are often not superior to bidirectional blind search (Kaindl and Kainz 1997; Barker and Korf 2015).

Discussion

We have seen a theoretical and empirical analysis of the search behavior of BDDA* and revealed a fundamental problem: the use of a heuristic does not always improve the search performance of BDDA*. In general, even the perfect heuristic can exponentially impair search performance. The most important finding is that good distance estimations are not the correct quantity to improve the search performance of symbolic heuristic search. This is due to the fact that the search effort of symbolic search is not directly related to the number of explicit states that have to be expanded. Therefore, it is unlikely that the concept of distance estimators is generally as helpful in symbolic search as it is in explicit search. The important question is: how can we use heuristics in symbolic search in a way that they usually pay off?

We believe that the overall goal is to keep the search effort, i.e., the cumulative size of the BDDs representing the states that BDDA* must expand, to determine an optimal plan, as small as possible. In BDDA*, a heuristic partitions a set of states into several sets of states. One possible idea is to see heuristics as a new flexible method of partitioning a set of states. Interestingly, partitioning has already been successfully applied to merge transition relations (Torralba, Edelkamp, and Kissmann 2013). Another direction is the use of heuristics and decision diagrams that provide size guarantees. For example, potential heuristics (Pommerening et al. 2015) always have a linear-size representation using edge-valued multi-valued decision diagrams.

Finally, it makes sense to consider heuristic search algorithms other than BDDA*. To capture the worst-case performance of bidirectional heuristic search, the idea of must-expand states was generalized to must-expand state pairs where at least one of the two states has to be expanded. Holte et al. (2016) have introduced a bidirectional heuristic search algorithm that always meets in the middle. Chen et al. (2017) presented Near-Optimal Bidirectional Search (NBS), which has the guarantee of having at most twice as many expansions as the minimum number of expansions necessary to cover all must-expand pairs. However, explicit A* and explicit NBS have guarantees on node expansions that represent single explicit states. Therefore, these results cannot be generalized for symbolic heuristic search and it is unlikely that a symbolic version of NBS has any guarantees highly relevant to symbolic search.

Conclusion

We theoretically and empirically evaluated the search behaviour of BDDA*. In general, we proved that the use of a heuristic does not always improve the search performance of BDDA* and can improve or impair search performance exponentially. The most important finding is that good distance estimations are not the correct quantity to improve the search performance of symbolic heuristic search. This shows that it is unlikely that the concept of distance estimators, i.e., heuristics, is generally as helpful in symbolic search as it is in explicit search. Possible other ways to use heuristics in symbolic search are to consider heuristics as new degrees of freedom that allow multiple different partitions of a set of states, or to use "size-aware" heuristics that provide guarantees which keep the representation size small.

Acknowledgments

David Speck was supported by the German Research Foundation (DFG) as part of the project EPSDAC (MA 7790/1-1). Florian Geißer was supported by ARC project DP180103446, *On-line planning for constrained autonomous agents in an uncertain world*. We thank Álvaro Torralba and the anonymous reviewers for their comments and suggestions.

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proc. ICAPS 2015*, 2–6.

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

Barker, J. K., and Korf, R. E. 2015. Limitations of front-to-end bidirectional heuristic search. In *Proc. AAAI 2015*, 1086–1092.

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

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-end bidirectional heuristic search with near-optimal node expansions. In *Proc. IJCAI 2017*, 489–495.

Edelkamp, S., and Kissmann, P. 2011. On the complexity of BDDs for state space search: A case study in Connect Four. In *Proc. AAAI 2011*, 18–23.

Edelkamp, S., and Reffel, F. 1998. OBDDs in heuristic search. In *Proc. KI 1998*, 81–92.

Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In *Proc. AIPS 2002*, 274–283.

Franco, S.; Torralba, Á.; Lelis, L. H. S.; and Barley, M. 2017. On creating complementary pattern databases. In *Proc. IJCAI 2017*, 4302–4309.

Franco, S.; Lelis, L. H. S.; Barley, M.; Edelkamp, S.; Martines, M.; and Moraru, I. 2018. The Complementary1 planner in the IPC 2018. In *IPC-9 planner abstracts*, 27–29.

Hansen, E. A.; Zhou, R.; and Feng, Z. 2002. Symbolic heuristic search using decision diagrams. In *Proc. SARA 2002*, 83–98.

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., and Röger, G. 2008. How good is almost perfect? In *Proc. AAAI 2008*, 944–949.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Holte, R. C.; Felner, A.; Sharon, G.; and Sturtevant, N. R. 2016. Bidirectional search that is guaranteed to meet in the middle. In *Proc. AAAI 2016*, 3411–3417.

Jensen, R. M.; Veloso, M. M.; and Bryant, R. E. 2008. State-set branching: Leveraging BDDs for heuristic search. *AIJ* 172(2-3):103–139.

Kaindl, H., and Kainz, G. 1997. Bidirectional heuristic search reconsidered. *JAIR* 7:283–317.

Kissmann, P., and Edelkamp, S. 2011. Improving cost-optimal domain-independent symbolic planning. In *Proc. AAAI 2011*, 992–997.

Kissmann, P., and Hoffmann, J. 2014. Bdd ordering heuristics for classical planning. *JAIR* 51:779–804.

Kissmann, P.; Edelkamp, S.; and Hoffmann, J. 2014. Gamer and dynamic-gamer – symbolic search at ipc 2014. In *IPC-8 planner abstracts*, 77–84.

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

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

Moraru, I.; Edelkamp, S.; Franco, S.; and Martinez, M. 2019. Simplifying automated pattern selection for planning with symbolic pattern databases. In *Proc. KI 2019*, 249–263.

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

Pohl, I. 1969. First results on the effect of error in heuristic search. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 5*. Edinburgh University Press. 219–236.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From non-negative to general operator cost partitioning. In *Proc. AAAI 2015*, 3335–3341.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated cost partitioning for optimal classical planning. *JAIR* 67:129–167.

Speck, D.; Geißer, F.; Mattmüller, R.; and Torralba, Á. 2019. Symbolic planning with axioms. In *Proc. ICAPS 2019*, 464–472.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018a. Symbolic planning with edge-valued multi-valued decision diagrams. In *Proc. ICAPS 2018*, 250–258.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018b. SYMPLE: Symbolic Planning based on EVMDDs. In *IPC-9 planner abstracts*, 82–85.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic top-k planning. In *Proc. AAAI 2020*. To appear.

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.

Torralba, Á.; Edelkamp, S.; and Kissmann, P. 2013. Transition trees for cost-optimal symbolic planning. In *Proc. ICAPS 2013*, 206–214.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2016. Abstraction heuristics for symbolic bidirectional search. In *Proc. IJCAI 2016*, 3272–3278.

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