# Abstraction Heuristics for Symbolic Bidirectional Search

**Álvaro Torralba**
Saarland University
Saarbrücken, Germany
torralba@cs.uni-saarland.de

**Carlos Linares López** and **Daniel Borrajo**
Universidad Carlos III de Madrid
Madrid, Spain
{carlos.linares,daniel.borrajo}@uc3m.es

## Abstract

Symbolic bidirectional uniform-cost search is a prominent technique for cost-optimal planning. Thus, the question whether it can be further improved by making use of heuristic functions raises naturally. However, the use of heuristics in bidirectional search does not always improve its performance. We propose a novel way to use abstraction heuristics in symbolic bidirectional search in which the search only resorts to heuristics when it becomes unfeasible. We adapt the definition of partial and perimeter abstractions to bidirectional search, where A$^*$ is used to traverse the abstract state spaces and/or generate the perimeter. The results show that abstraction heuristics can further improve symbolic bidirectional search in some domains. In fact, the resulting planner, SymBA$^*$, was the winner of the optimal-track of the last IPC.

## 1 Introduction

Most cost-optimal planners are based on A$^*$ guided with an admissible heuristic. Bidirectional search has not been explored so extensively, due to the inherent difficulties of regression in planning and the computational cost of detecting collisions between both frontiers [Alcázar *et al.*, 2014]. Symbolic search [McMillan, 1993] reasons over sets of states, substantially reducing the cost of detecting the collision of frontiers. Besides, recent advances have alleviated the problem of spurious states in symbolic regression [Torralba and Alcázar, 2013]. Thus, symbolic bidirectional uniform-cost search (SB) is among the best algorithms for cost-optimal planning, outperforming not only A$^*$-based planners but also BDDA$^*$, the symbolic search variant of A$^*$.

These observations lead to the question of whether heuristics can further improve SB. Bidirectional heuristic search (BHS) has a long history [Pohl, 1969; de Champeaux, 1983], but it has never convincingly outperformed A$^*$ across a significant number of domains. There have been various attempts to explain the main reasons behind the limitations of front-to-end BHS, from the search frontiers passing each other without intersecting [Nilsson, 1982] to the hardness of proving optimality [Kaindl and Kainz, 1997]. A recent study conjectures that the quality of heuristics is a major factor for explaining the disappointing empirical results of BHS [Barker and Korf, 2015] and motivating new approaches [Holte *et al.*, 2016].

Perimeter search is a variant of BHS that creates a perimeter around the goal, and uses heuristics that estimate the distance to the perimeter instead of to the goal [Dillenburg and Nelson, 1994]. Abstraction heuristics are a good fit because they precompute the heuristic, avoiding a large overhead during the search [Eyerich and Helmert, 2013]. Moreover, symbolic perimeter abstraction heuristics are state-of-the-art for cost-optimal planning [Torralba *et al.*, 2013].

We present a new planner, SymBA$^*$, that combines symbolic bidirectional search with perimeter abstraction heuristics, exploiting their synergy to benefit from the advantages of BHS and overcome its limitations. SymBA$^*$ performs bidirectional searches over different state spaces. It starts in the original search space and, when the search becomes too hard, it derives a perimeter abstraction heuristic. The planner decides at any point whether to advance the search in the original state space, enlarging the perimeter, or in an abstract space to improve the heuristic. To that end, we introduce a new type of abstraction heuristics that uses bidirectional search combined with perimeter and partial abstractions. This is the first time bidirectional search is used to explore abstract state spaces to the best of the authors' knowledge. Even though the theory behind partial and perimeter abstractions has been well studied, they have to be adapted for their combination with bidirectional search. In particular, we study how partial abstractions can be used when A$^*$ search is used to traverse the abstract state space and how the initialization of perimeter abstractions can be improved in the bidirectional setting.

## 2 Preliminaries

A *planning task* is a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$. $\mathcal{V}$ is a finite set of *variables* $v$, each $v \in \mathcal{V}$ being associated with a finite domain $D_v$. A *partial state* over $\mathcal{V}$ is a function $s$ on a subset $V(s)$ of $\mathcal{V}$, so that $s(v) \in D_v$ for all $v \in V(s)$; $s$ is a *state* if $V(s) = \mathcal{V}$. The *initial state* $\mathcal{I}$ is a state. The *goal* $\mathcal{G}$ is a partial state. $\mathcal{A}$ is a finite set of *actions*, each $a \in \mathcal{A}$ being a pair $(pre_a, \mathit{eff}_a)$ of partial states, called its *precondition* and *effect*. Each $a \in \mathcal{A}$ has a non-negative *cost*, $c(a) \in \mathbb{R}_0^+$.

The *state space* of a planning task $\Pi$ is a labeled transition system $\Theta^\Pi = (S, L, T, s_0, S_G)$ where: $S$ is the set of all states; $s_0$ is the initial state $\mathcal{I}$ of $\Pi$; $s \in S_G$ iff $\mathcal{G} \subseteq s$; the

labels $L$ correspond to the actions $A$, and $s \xrightarrow{a} t$ is a transition in $T$ if $s$ complies with $pre_a$, and $t(v) = \mathit{eff}_a(v)$ for $v \in V(\mathit{eff}_a)$ while $t(v) = s(v)$ for $v \in V \setminus V(\mathit{eff}_a)$. A *plan* for $s$ is a path from $s$ to any $s_G \in S_G$. The cost of a plan is defined as $c(\pi) = \sum_{a_i \in \pi} c(a_i)$. The cost of a cheapest plan for $s$ is denoted $h^*(s)$ and the cost of the cheapest path from $s_0$ to $s$ is denoted $g^*(s)$. A plan for $s_0$ is a plan for $\Pi$, and is *optimal* iff its cost equals $h^*(s_0)$.

Most cost-optimal planners use heuristic search with A$^*$. A *heuristic* is a function $h : \mathcal{S} \to \mathbb{R}_0^+ \cup \{\infty\}$ which estimates the remaining cost to reach the goal. A heuristic is *perfect* if it coincides with $h^*$, and it is *admissible* if it never overestimates the optimal cost, that is, $\forall s : h(s) \leq h^*(s)$. A heuristic is *consistent* if for every transition $s \xrightarrow{a} t$, $h(s) \leq h(t) + c(a)$. A$^*$ expands the nodes with lowest $f(s) = g(s) + h(s)$ first so that no node with $f(s) > h^*(\mathcal{I})$ is ever expanded. If the heuristic is admissible A$^*$ is guaranteed to return an optimal solution. Moreover, if the heuristic is consistent, A$^*$ always closes states with their optimal $g$-value, $g^*(s)$, so it does not re-expand any node.

## 2.1 Bidirectional Search

A bidirectional search, $\mathcal{T}$ is composed of a forward, $\mathcal{T}_{fw}$, and a backward, $\mathcal{T}_{bw}$, unidirectional search. We use $\mathcal{T}_u$ to denote a unidirectional search in an unspecified direction and $\mathcal{T}_{\neg u}$ for the search in the opposite direction. Each search $\mathcal{T}_u$ consists of an open, $\mathit{open}(\mathcal{T}_u)$, and a closed list, $\mathit{closed}(\mathcal{T}_u)$. We denote by $g(\mathcal{T}_u)$ and $f(\mathcal{T}_u)$ the minimum $g$ and $f$-values of any state in $\mathit{open}(\mathcal{T}_u)$, respectively.

*Nipping* is an optimization that avoids expanding any state if it has already been expanded in the opposite direction [Kwa, 1989]. We apply nipping at generation time in order to avoid unnecessary evaluations. We rely on states always being closed with their optimal $g$-value, $g^*$, so that $g^*(s)$ is known for all $s \in \mathit{closed}(\mathcal{T}_u)$ and those $s \in \mathit{open}(\mathcal{T}_u)$ with $g^*(s) \leq g(\mathcal{T}_u) + \min_{a \in \mathcal{A}} c(a)$. Thus, whenever a state $s$ is generated or is going to be expanded in $\mathcal{T}_u$, if $g^*(s)$ is known for $\mathcal{T}_{\neg u}$, we discard $s$ (without introducing it in $\mathit{closed}(\mathcal{T}_u)$) and store the plan through that state if it is the best plan found so far. The algorithm terminates when the best solution found so far, $\pi$, has been proven optimal, i. e., $c(\pi) \leq \max(f(\mathcal{T}_{fw}), f(\mathcal{T}_{bw}))$. Both frontiers use each other as a heuristic, assigning an admissible value of $g(\mathcal{T}_{\neg u}) + \min_{a \in \mathcal{A}} c(a)$ to all states in $\mathit{open}(\mathcal{T}_u)$. As all states in $\mathit{open}(\mathcal{T}_u)$ have the same heuristic value, this does not affect to the expansion order of uniform-cost search, but allows to terminate the algorithm whenever $c(\pi) \leq g(\mathcal{T}_{bw}) + g(\mathcal{T}_{fw}) + \min_{a \in \mathcal{A}} c(a)$.

## 2.2 Abstraction Heuristics

An abstraction is a mapping $\alpha : \mathcal{S} \to \mathcal{S}^\alpha$ from states to abstract states. The *abstract state space* is a tuple $\Theta^\alpha = \langle \mathcal{S}^\alpha, L, T^\alpha, \mathcal{I}^\alpha, \mathcal{S}^\alpha_\star \rangle$ where $\mathcal{S}^\alpha$ is the set of abstract states, $L$ is the set of labels, $T^\alpha = \{(\alpha(s) \xrightarrow{a} \alpha(t)) \mid s \xrightarrow{a} t\}$, $\mathcal{I}^\alpha = \alpha(s_0)$ and $\mathcal{S}^\alpha_\star = \{s^\alpha \mid \exists s \in \mathcal{S}_\star, s^\alpha = \alpha(s)\}$. $\mathcal{T}^\alpha$ denotes a bidirectional search in $\Theta^\alpha$, in contrast to the search in the original state space, $\mathcal{T}^\Pi$. Abstraction heuristics use the optimal solution cost in $\Theta^\alpha$, $h^\alpha$, as an admissible estimation. Similarly, $g^\alpha(s)$ is the optimal solution cost from $\mathcal{I}^\alpha$ to $\alpha(s)$.

There are different types of abstraction heuristics depending on the mapping definition, $\alpha$. Pattern Databases (PDBs) [Culberson and Schaeffer, 1998; Edelkamp, 2001] are projections of $\Pi$ onto a subset of variables (called *pattern*), so that two states are equivalent iff they agree on the value of variables in the pattern. Merge-and-shrink (M&S) abstractions generalize PDBs, allowing abstractions that use all variables [Helmert *et al.*, 2007; 2014].

The optimal solution cost from every abstract state, $h^\alpha(s^\alpha)$, is precomputed and stored in a lookup table prior to the search by performing a backward uniform-cost search from the abstract goal, $\mathcal{T}^\alpha_{bw}$. Partial abstractions do not search the entire abstract state space completely [Anderson *et al.*, 2007; Edelkamp and Kissmann, 2008b]. Thus, $h^\alpha$ is only known for states that were expanded during the precomputation phase or were left in open with $g$-value lower or equal than $g(\mathcal{T}^\alpha) + \min_{a \in \mathcal{A}} c(a)$. For every other abstract state, the heuristic returns the minimum cost with which a state could be generated $g(\mathcal{T}^\alpha) + \min_{a \in \mathcal{A}} c(a)$. Partial abstractions are admissible and consistent.

Perimeter abstractions construct a perimeter around the goal in the original state space and use it to seed the search in the abstract state space [Felner and Ofek, 2007; Eyerich and Helmert, 2013]. The perimeter is constructed by a backward search, $\mathcal{T}^\Pi_{bw}$, which computes the perfect heuristic for all states in $\mathit{closed}(\mathcal{T}^\Pi_{bw})$. For every state outside the perimeter, an abstract search, $\mathcal{T}^\alpha_{bw}$ computes the minimum distance from each abstract state to the closest abstract state in the perimeter. Formally, $\mathcal{T}^\alpha_{bw}$ is initialized with: $\mathit{open}(\mathcal{T}^\alpha_{bw})[g] = \{s^\alpha \mid \exists_{s \in S, \alpha(s) = s^\alpha} \; s \in \mathit{open}(\mathcal{T}^\Pi_{bw})[g]\}$ and $\mathit{closed}(\mathcal{T}^\alpha_{bw}) = \{s^\alpha \mid \forall_{s \in S, s^\alpha = \alpha(s)} \; s \in \mathit{closed}(\mathcal{T}^\Pi_{bw})\}$.

## 2.3 Symbolic Search

Symbolic search algorithms use succinct data-structures like Binary Decision Diagrams [Bryant, 1986] to efficiently represent and manipulate sets of states. BDDs offer a compact representation of sets of states that sometimes can get an exponential advantage in memory with respect to their explicit enumeration [Edelkamp and Kissmann, 2008a]. Furthermore, BDD operations can be used to compute the union or intersection of two sets of states. Using these operations, it is possible to define symbolic versions of different search algorithms like uniform-cost search or A$^*$.

BDDA$^*$ is the symbolic version of A$^*$. As usual, it expands states in ascending order of their $f$-value, but expanding at the same time all the states sharing the same $f$ and $g$ values. A difference with typical explicit implementations of A$^*$ is that it uses the opposite tie-breaking. In BDDA$^*$ states with lower $g$-value are preferred in order to generate all the states with the same $f$ and $g$ value before expanding any of them.

Symbolic search is not limited to the original state space. Symbolic PDBs take advantage of symbolic search in order to traverse the abstract state space. This allows for the use of larger patterns, since the state space is not explicitly enumerated [Kissmann, 2012]. The combination of symbolic search and perimeter abstractions is a state-of-the-art heuristic [Torralba *et al.*, 2013], which we extend to the bidirectional case.

## 3 SymBA*: Symbolic Bidirectional A*

SymBA* performs several symbolic bidirectional A* searches on different state spaces. First, SymBA* starts a bidirectional search in the original state space, $\mathcal{T}^{\Pi}$. At each iteration, the algorithm performs a step in a selected direction, i. e., expands the set of states with minimum $f$-value in the frontier. Since no abstraction heuristic has been derived yet, it behaves like symbolic bidirectional uniform-cost search. This search continues until the next layer in both directions is deemed as unfeasible, because SymBA* estimates that it will take either too much time or memory. Only then, a new bidirectional search is started in an abstract state space, $\mathcal{T}^{\alpha}$ initialized with the current frontiers of $\mathcal{T}^{\Pi}$. The abstract searches provide heuristic estimations, increasing the $f$-value of states in the original search frontiers. Eventually, the search in the original state space will be simplified (as the number of states with minimum $f$-value will be smaller)[1] and SymBA* will continue the search in the original state space.

One important feature of the algorithm is the lazy evaluation of the heuristics. The search in abstract state spaces is delayed until strictly needed to simplify the original search, allowing SymBA* to use multiple abstraction heuristics without a large overhead. We model this by considering a pool of active searches and letting the algorithm decide which search should be advanced at any step, as shown in Alg. 1. The pool of searches is initialized with a bidirectional search in the original state space. At each iteration, the algorithm filters the searches that are *valid candidates* from the pool and selects the most promising one. The algorithm depends on this *search selection strategy*, further explained in Section 7.

Once a search has been selected, the procedure `ExpandFrontier` expands the set of states that have a minimum $g$-value among those that have a minimum $f$-value, as usual in BDDA*. If we progress the original search, a new plan with a lower cost than the incumbent solution may be found. If an abstract search is selected, we update the heuristic value of states in searches in the opposite direction in the pool, both in the abstract and original state spaces. If several abstraction heuristics are generated, SymBA* will use their maximum value, so that the heuristic value of states can only be increased. If there are no valid search candidates (line 10), a new bidirectional search is added to the pool (which amounts to two new searches). The *abstraction strategy* relaxes the current frontiers of the original search, until the frontier size is small enough to continue the search.

One of the main characteristics of SymBA* is that the heuristics change dynamically during the search. Not only may the algorithm decide to initialize a new abstract search at any point, but also every time that an abstract search performs a step, the heuristic value of states in the original search may increase. Re-evaluating the entire search frontier repeatedly may be too costly if done naïvely, becoming a bottleneck and making the entire algorithm unfeasible. We avoid this problem using the lazy implementation of BDDA* [Edelkamp *et al.*, 2012], which keeps the states organized by $g$-value and defers the heuristic evaluation. Thus, whenever the heuristic changes in the middle of the search only the set of states currently selected for expansion must be re-evaluated, without any additional computation in the open list.

[1] Having fewer states does not necessarily imply that the BDD is smaller, but in most cases there is a positive correlation.

**Algorithm 1:** SymBA*

**Input**: Planning problem: $\Pi = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$
**Output**: Cost-optimal plan or "no plan"

1 $SearchPool \leftarrow \{\mathcal{T}^{\Pi}_{fw}, \mathcal{T}^{\Pi}_{bw}\}$
2 $\pi \leftarrow$ "$noplan$"
3 **while** $\max(f(\mathcal{T}^{\Pi}_{fw}), f(\mathcal{T}^{\Pi}_{bw})) < cost(\pi)$ **do**
4  **if** $\exists \mathcal{T}^{X}_{u} \in SearchPool$ s.t. `Is-Candidate(`$\mathcal{T}^{X}_{u}$`)` **then**
5  $\mathcal{T}^{X}_{u} \leftarrow$ `Select-Search(`$SearchPool$`)`
6  $\pi' \leftarrow$ `Expand-frontier(`$\mathcal{T}^{X}_{u}$`)`
7  **if** $X = \Pi \wedge \pi' \neq \emptyset \wedge cost(\pi') < cost(\pi)$ **then**
8  $\pi \leftarrow \pi'$
9  `Notify-h(`$\mathcal{T}^{X}_{\neg u}, \mathcal{T}^{\Pi}_{\neg u}$`)`
10  **else**
11  $\alpha \leftarrow$`Select-abstraction(`$\Pi, \mathcal{T}^{\Pi}_{fw}, \mathcal{T}^{\Pi}_{bw}$`)`
12  $\left\langle \mathcal{T}^{\alpha}_{fw}, \mathcal{T}^{\alpha}_{bw} \right\rangle \leftarrow$`Apply(`$\alpha, \mathcal{T}^{\Pi}_{fw}, \mathcal{T}^{\Pi}_{bw}$`)`
13  $SearchPool \leftarrow SearchPool \cup \{\mathcal{T}^{\alpha}_{fw}, \mathcal{T}^{\alpha}_{bw}\}$
14 **return** $\pi$

The next sections describe the abstraction heuristics that we use. Summarizing, (i) bidirectional search can be used in the abstract state space allowing two searches, in opposite directions, to exchange information and avoid redundant work; (ii) partial abstractions allow SymBA* to traverse larger abstract state spaces with less effort, since exploring them completely is unnecessarily expensive; and (iii) perimeter abstractions take advantage of the searches in the original state space in order to obtain better estimates, overcoming the limitations of front-to-end BHS. Perimeter and partial abstraction heuristics are not new and the conditions for consistency and admissibility are well-known for them. However, the use of bidirectional abstract searches and, in particular, the use of A* searches for exploring the abstract state space and constructing the perimeter, requires us to reconsider partial and perimeter abstractions. Our aim is to obtain heuristic estimations as informed as possible while preserving the optimality of the algorithm. Inconsistency of heuristics is not necessarily a problem [Felner *et al.*, 2011], but states must be closed with their optimal $g^*$-value to ensure that perimeter abstractions are admissible.

## 4 Bidirectional Abstractions

To perform bidirectional search in abstract state spaces, a distinction must be made between the searches used in the original and abstract state spaces. $\mathcal{T}^{\Pi}$ aims to find a plan. So, whenever the two frontiers meet, a plan is retrieved and nipping avoids the expansion of the state to eliminate redundant work between $\mathcal{T}^{\Pi}_{fw}$ and $\mathcal{T}^{\Pi}_{bw}$ . On the other hand, searches on abstract state spaces are used to derive heuristic estimates for the original search. Therefore, nipping must be disabled in order for the estimations to be admissible.

Thus, the interaction between both searches is reduced to use each other as a (perfect) heuristic. But, otherwise, they do not directly interact to detect the collision of their frontiers. Hence, bidirectional searches in the abstract state space are two A$^*$ searches using each other as a heuristic. They must redundantly expand states that have already been expanded in the other direction in order to provide admissible estimations to the original search. In the worst case, if both abstract searches traverse the entire abstract state space, the search effort is doubled plus an overhead for using heuristics. Here it is where partial abstractions come in handy to avoid the exploration of the entire abstract state space.

## 5 Partial Abstractions with Heuristic Search

SymBA$^*$ uses partial abstractions to traverse large abstract state spaces that could not be entirely explored otherwise. In order to compute the heuristic value of every state, we will likely need to expand most parts of the abstract state space. For example, if one single state is a dead-end both in the original and the abstract state space, the abstract state space must be completely traversed before continuing the search. Most parts of that computation are irrelevant, since the termination criterion of A$^*$ is that, for every state $s$ not expanded yet, $f^*(s) \geq c(\pi)$. In other words, the heuristic value of a state does not matter provided it is large enough to guarantee that its $f$-value is not the minimum among the current $f$-value of other states in the search. To simplify the notation, we assume wlog (the same arguments hold for opposite directions) that a backward abstract A$^*$ search, $\mathcal{T}^\alpha_{bw}$ guided with an admissible estimation of $g^\alpha$ is used to compute $h^\alpha$ in order to inform a forward search in the original state space, $\mathcal{T}^\Pi_{fw}$.

In partial abstractions, abstract states can be classified depending on whether $h^\alpha(s)$ is known or not. $h^\alpha(s)$ is known for those abstract states that have already been expanded and those that remain in open with a $g$-value lower or equal than $g(\mathcal{T}^\alpha_{bw}) + \min_{a\in\mathcal{A}} c(a)$. The question is which heuristic value do we assign to states for which $h^\alpha(s)$ is not known. The usual answer is to use the minimum $g$-value with which they could be generated, $g(\mathcal{T}^\alpha_{bw}) + \min_{a\in\mathcal{A}} c(a)$. Usually, this is a satisfactory lower bound because the abstract state space is explored with a uniform-cost search so $g(\mathcal{T}^\alpha_{bw})$ is constantly increasing. However, in SymBA$^*$ the abstract state space is explored with an A$^*$ search because it uses the other frontier as (perfect) heuristic. Thus, a bound only based on $g(\mathcal{T}^\alpha_{bw})$ is no longer useful, because it may remain constant as the search progresses. Therefore, states, we set a bound for the $f$-value of the states, based on the following inequality: $f(\mathcal{T}^\alpha_{bw}) \leq f^\alpha(s) \leq f^*(s)$. As $f(s) = g(s) + h(s)$, we can translate the bound for $f(s)$ to a heuristic value that depends on the $g$-value of $s$: $h(s) = f(\mathcal{T}^\alpha_{bw}) - g(s)$.

**Proposition 1.** *Let $\mathcal{T}^\alpha_{bw}$ be a search in $\Theta^\alpha$ and $s \in S$ be a state s.t. $\alpha(s) \notin closed(\mathcal{T}^\alpha_{bw})$. Then $f(\mathcal{T}^\alpha_{bw}) - g(s) \leq h^*(s)$.*

*Proof.* By definition $g(s) \geq g^*(s)$. Also, $f(\mathcal{T}^\alpha_{bw}) \leq f^\alpha(s) \leq f^*(s) = g^*(s) + h^*(s)$. Thus, the inequality holds: $h^*(s) = f^*(s) - g^*(s) \geq f^*(s) - g(s) \geq f(\mathcal{T}^\alpha_{bw}) - g(s)$. □

The problem is that such heuristic may be inconsistent, as illustrated by Figure 1. Consistency requires that for every $s \xrightarrow{a} t$, $h(s) \leq h(t) + c(a)$. However, if $\alpha(s)$ has been expanded by the abstract search and $\alpha(t)$ not, the bound for $\alpha(t)$, $f(\mathcal{T}^\alpha_{bw}) - g(t)$ might be a lot weaker than $h^\alpha(s)$, generating an inconsistency. This may cause $t$ to be expanded before $s$ with a suboptimal $g$-value. In order to avoid closing any state with a suboptimal $g$-value, one must ensure that the lower-bound $h^\alpha(s)$ is only used when $f(\mathcal{T}^\alpha_{bw})$ is large enough to satisfy $h^\alpha(s) \leq f(\mathcal{T}^\alpha_{bw}) - g(t)$. So, the abstract search cannot be stopped at any point, it must continue until $f(\mathcal{T}^\Pi_{fw}) \leq f(\mathcal{T}^\alpha_{bw})$. We model this by computing the minimum between $h^\alpha$ and our $f$-based bound.

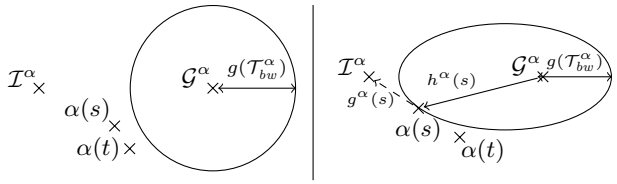


Figure 1: $\mathcal{T}^\alpha_{bw}$ with uniform-cost search (left) and A$^*$ (right).

**Definition 1** (Partial abstraction heuristic)**.** *Let $\mathcal{T}^\alpha_{bw}$ be an A$^*$ search over $\Theta^\alpha$ informed with an admissible and consistent heuristic. We define the bound for unexplored states $B$, as $B(s, \mathcal{T}^\alpha_{bw}) := \max\{f(\mathcal{T}^\alpha_{bw}) - g(s), g(\mathcal{T}^\alpha_{bw}) + \min_{a\in\mathcal{A}} c(a)\}$. We define the* partial abstraction heuristic *as:*

$$h_{\mathcal{T}^\alpha}(s) = \begin{cases} \min(h^\alpha(s), B(s, \mathcal{T}^\alpha_{bw})) & \text{if } h^\alpha(s) \text{ is known in } \mathcal{T}^\alpha_{bw} \\ B(s, \mathcal{T}^\alpha_{bw}) & \text{otherwise} \end{cases}$$

$h_{\mathcal{T}^\alpha}$ is still an inconsistent heuristic because if neither $h^\alpha(s)$ nor $h^\alpha(t)$ are known, $B(s, \mathcal{T}^\alpha_{bw}) = f(\mathcal{T}^\alpha_{bw}) - g(s)$ and $B(t, \mathcal{T}^\alpha_{bw}) = f(\mathcal{T}^\alpha_{bw}) - g(t)$ then $h(s) = f(\mathcal{T}^\alpha_{bw}) - g(s)$ and $h(t) = f(\mathcal{T}^\alpha_{bw}) - g(t)$. This is clearly inconsistent since it may be that $g(s) \ll g(t)$, since $g^*(t) \leq g^*(s) + c(a)$ but $s$ has not been expanded yet so $g^*(t) \ll g(t)$ is possible. However, then $f(s) = f(t)$, and the one with lower $g$-value is selected for expansion. Theorem 2 proves this in general.

**Theorem 2.** *Let $\mathcal{T}^\Pi_{fw}$ be an A$^*$ search that breaks ties in favor of states with lower g-value*[2] *and is informed with a $h_{\mathcal{T}^\alpha}$. Then, for any $s \in closed(\mathcal{T}^\Pi_{fw}), g(s) = g^*(s)$.*

*Proof Sketch.* (Full proof in TR) Suppose that $g(s) > g^*(s)$. By Lemma 1 in [Hart *et al.*, 1968], there exists a state $r \in open(\mathcal{T}^\Pi_{fw})$, $r \neq s$, which is in the optimal path from $\mathcal{I}$ to $s$ such that $g(r) = g^*(r) \leq g^*(s) < g(s)$. Since $s$ was selected for expansion, $f(s) < f(r)$. This leads to contradiction for any possible values of $h_{\mathcal{T}^\alpha}(s)$ (and $h_{\mathcal{T}^\alpha}(r)$):

- Assume $h_{\mathcal{T}^\alpha}(s) = h^\alpha(s)$. As $h_{\mathcal{T}^\alpha}(r) \leq h^\alpha(r)$, this leads to contradiction with the consistency of $h^\alpha$.
- Assume $h_{\mathcal{T}^\alpha}(s) = f(\mathcal{T}^\alpha_{bw}) - g(s)$. As $g(s) \geq g^*(r) = g(r)$, it follows that $B(r, \mathcal{T}^\alpha_{bw}) = f(\mathcal{T}^\alpha_{bw}) - g(r)$. Therefore, $f(r) \leq f(\mathcal{T}^\alpha_{bw}) = f(s)$ with contradiction.
- Assume $h_{\mathcal{T}^\alpha}(s) = g(\mathcal{T}^\alpha_{bw}) + \min_{a\in\mathcal{A}} c(a)$. As $g(s) \geq g^*(r) = g(r)$, it follows that $B(r, \mathcal{T}^\alpha_{bw}) = g(\mathcal{T}^\alpha_{bw}) + \min_{a\in\mathcal{A}} c(a)$. Hence, $h(r) \leq f(s)$ with contradiction.

□

[2]The tie-breaking condition is not needed if $f(\mathcal{T}^\alpha_{bw}) > f(\mathcal{T}^\Pi_{fw})$.

## 6 Perimeter Bidirectional Abstractions

Perimeter abstractions must also be redefined to handle bidirectional and heuristic searches appropriately. First, in bidirectional search we have forward and backward perimeters. This can be leveraged in the initialization of the abstract searches by ignoring states in both closed lists. The abstract bidirectional search $\mathcal{T}^\alpha$ is initialized with the perimeter of $\mathcal{T}^\Pi$ as: $open(\mathcal{T}^\alpha_u)[g] = \{s^\alpha_j \mid \exists_{s\in S}\ \alpha(s) = s^\alpha_j \wedge s \in open(\mathcal{T}^\Pi_u)[g]\}$ and $closed(\mathcal{T}^\alpha_u) = \{s^\alpha_j \mid \forall_{s\in S}\ \alpha(s) = s^\alpha_j \implies s \in closed(\mathcal{T}^\Pi_{fw}) \cup closed(\mathcal{T}^\Pi_{bw})\}$.

Second, the perimeter search is carried out with A$^*$ instead of uniform-cost search. Hence, the perimeter is not uniform, i. e., it has not only expanded all states up to a fixed radius. The resulting heuristic is still admissible as long as every state in the perimeter is closed with its optimal value, $g^*(s)$ in $\mathcal{T}^\Pi_{fw}$ and $h^*(s)$ in $\mathcal{T}^\Pi_{bw}$. However, non-uniform perimeters may cause the heuristic to be inconsistent. For example, consider two states $s, t$ such that $s \xrightarrow{a} t$, $s \in closed(\mathcal{T}^\Pi_{bw})$ and $t \notin closed(\mathcal{T}^\Pi_{bw})$. In this case, $h(s) = h^*(s)$ and $h(t) = h^\alpha(t)$, so consistency may be violated: $h(s) \not\leq h(t) + c(a)$. However, this poses no problem if nipping is used to eliminate all states in the opposite perimeter because the heuristic will never be evaluated on those states anyway. Theorem 3 proves that the perimeter abstraction heuristic is still admissible and consistent for all relevant states, i. e., those that are not pruned by nipping.

**Theorem 3.** *Let $\mathcal{T}^\alpha_u$ be a perimeter abstraction in $\Theta^\alpha$ initialized with the perimeters of $\mathcal{T}^\Pi$. Then, $h$ is admissible and consistent for any state $s \notin closed(\mathcal{T}^\Pi_{fw}) \cup closed(\mathcal{T}^\Pi_{bw})$.*

*Proof.* A heuristic $h$ is consistent if and only if $h(s) \leq h(t) + c(a)\ \forall s \xrightarrow{a} t$. In our case, we only contemplate states $s, t \notin closed(\mathcal{T}^\Pi_{fw}) \cup closed(\mathcal{T}^\Pi_{bw})$, so $\alpha(s)$ and $\alpha(t)$ will not be introduced in $closed(\mathcal{T}^\alpha_u)$ when initializing the abstract search. $\alpha(t)$ may be generated by $\mathcal{T}^\alpha_u$ or not. If not, $h(t) = \infty$ and consistency follows. If $\alpha(t)$ is expanded by $\mathcal{T}^\alpha_u$, $\alpha(s)$ will be inserted in $open(\mathcal{T}^\alpha_u)$ and later expanded. Hence, by consistency of $h^\alpha$, it follows that $h(s) \leq h(t) + c(a)$. □

Therefore, bidirectional A$^*$ guided with perimeter abstraction heuristics returns the optimal solution since the heuristic is consistent for every state outside the perimeter and nipping prevents its evaluation in states in the perimeter.

## 7 Search Selection Strategy

The core of SymBA$^*$, as outlined in Algorithm 1, is how to decide which search should be pushed forward at each iteration. A search is a valid candidate if and only if it is both *feasible* and *useful*. A search is *feasible* if the estimated time and number of nodes needed to perform the next step does not surpass any of the bounds imposed as parameters. The search in the original search space is always *useful*. A search in an abstract search space is *useful* if and only if it can further inform the next layer in the original search space, i.e., it has the potential of simplifying the search in the original state space by changing which states are selected for expansion.

**Definition 2** (Useful abstract search)**.** *Let $\mathcal{T}^\Pi_u$ be an A$^*$ search over $\Theta^\Pi$ and let $\mathcal{T}^\alpha_{\neg u}$ be an abstract search over $\Theta^\alpha$ in the opposite direction. Let $S_f$ be the set of states currently selected for expansion in $\mathcal{T}^\Pi_u$, i. e., a subset of those with minimal $f$-value according to any given tie-breaking criteria. We say that $\mathcal{T}^\alpha_{\neg u}$ is useful for $\mathcal{T}^\Pi_u$ if $f(\mathcal{T}^\alpha_{\neg u}) \leq f(\mathcal{T}^\Pi_u)$ and $\exists_{s\in S_f} h^\alpha(s)$ is not known or $B(s, \mathcal{T}^\alpha_{\neg u}) < h^\alpha(s)$.*

**Theorem 4.** *Let $\mathcal{T}^\Pi_u$ be an A$^*$ search informed with a heuristic generated by an abstract search $\mathcal{T}^\alpha_{\neg u}$. If $\mathcal{T}^\alpha_{\neg u}$ is not useful for $\mathcal{T}_u$, continuing the abstract search cannot alter the set of states selected for expansion, $S_f$.*

*Proof Sketch.* (Full proof in TR) Continuing $\mathcal{T}^\alpha_{\neg u}$ can only increase the $h$-value of the states. Therefore, the only way to alter $S_f$ is to increase the heuristic value of some $s \in S_f$. Since $h^\alpha(s) \leq B(s, \mathcal{T}^\alpha_{\neg u})$, $h(s)$ can only be increased if $h(s) < h^\alpha(s)$, which cannot be true if the search is not useful.

- If $f(\mathcal{T}^\Pi_u) = f(s) < f(\mathcal{T}^\alpha_{\neg u})$ then $h_{\mathcal{T}^\alpha}(s) < B(s, \mathcal{T}^\alpha_{\neg u})$ and $h(s) = h^\alpha(s)$.
- If $h^\alpha(s)$ is known and $h^\alpha(s) \leq B(s, T^\alpha_{\neg u})$ then $h(s) = h^\alpha(s)$.

□

The search in the original state space is preferred whenever it is feasible. Otherwise, among all the abstract searches that are *valid candidates*, we prefer those that have a greater minimum $f$-value, just because they are closer to proving that the current solution is optimal. In case of a tie, the search whose next step is expected to take less time is selected.

In summary, in order to prove optimality, it is enough to expand abstract searches until $f(\mathcal{T}^\alpha) \geq h^*(\mathcal{I})$. All the states whose abstract counterparts have not been expanded in any of the abstract searches do not need to be explored because their $f$-value is guaranteed to be non-optimal.

## 8 Experiments

SymBA$^*$ is implemented on top of Fast Downward [Helmert, 2006] and uses $h^2$ in a precomputation step to remove irrelevant actions and obtain mutex constraints for pruning the symbolic search [Alcázar and Torralba, 2015]. For our experiments we used the version of SymBA$^*$ that was submitted to IPC'14 after fixing some bugs. We run experiments on the optimal-track STRIPS planning instances from IPC'98 until IPC'14. All experiments were conducted on a cluster of Intel E5-2660 machines running at 2.20 GHz, with time (memory) cut-offs of 30 minutes (4 GB).

We consider a search feasible if the frontier has less than 10 million BDD nodes and each step takes less than 45 seconds, which are adequate values for the memory and time limits of our experiments. SymBA$^*$ can use any abstraction function that can be efficiently represented as BDDs such as PDBs and M&S with linear merge strategies [Edelkamp *et al.*, 2012; Helmert *et al.*, 2015]. In our evaluation we focus on the simpler variant, PDBs. To select the "pattern" of the PDBs we follow a strategy previously used for symbolic perimeter abstractions [Torralba *et al.*, 2013], which selects a variable ordering and relax variables one by one until the search can be

| | SB | SymBA* | | | | | | | | | | | Metis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PDB | | | | | | ipc1 | ipc2 | | | ∅ | |
| | | cgl | rev | lev | rnd | gcl | cgr | | ¬P | ¬B | | | |
| Airport(50) | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | 27 | **29** |
| Barman(34) | **18** | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | **18** | 11 |
| Blocks(35) | 31 | 31 | 31 | 31 | 31 | **32** | **32** | 31 | 31 | 30 | 31 | 30 | 28 |
| Childsnk(20) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | **6** |
| Depot(22) | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | **9** |
| Driverlog(20) | 12 | **14** | 13 | 13 | **14** | 13 | **14** | **14** | **14** | **14** | **14** | 12 | **14** |
| Elevators(50) | **44** | **44** | **44** | **44** | **44** | **44** | **44** | 43 | 43 | **44** | **44** | 43 | 40 |
| Floortile(40) | **34** | **34** | **34** | **34** | **34** | **34** | **34** | **34** | **34** | **34** | **34** | **34** | 16 |
| FreeCell(80) | 23 | 22 | 21 | 23 | 23 | **26** | 25 | 25 | 23 | 21 | 25 | 21 | 15 |
| GED(20) | **20** | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | **20** | 19 | 15 |
| Grid(5) | **3** | **3** | **3** | **3** | **3** | **3** | **3** | **3** | **3** | **3** | **3** | 2 | 2 |
| Hiking(20) | 15 | 15 | 15 | 15 | 19 | 18 | 18 | **20** | **20** | 18 | **20** | 15 | 14 |
| Logistics(63) | 23 | 25 | 25 | 25 | 25 | 25 | 24 | 25 | 25 | 25 | 25 | 23 | **27** |
| Miconic(150) | 112 | 107 | 108 | 108 | 109 | 108 | 108 | 108 | 108 | 108 | 108 | 108 | **144** |
| Mprime(35) | 24 | 23 | **25** | 24 | **25** | **25** | **25** | **25** | 24 | **25** | 24 | 23 | 24 |
| Mystery(30) | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | **18** |
| NoMyste(20) | 14 | 14 | 14 | **17** | 14 | 16 | 14 | 15 | 15 | 14 | **17** | 14 | **17** |
| Openstk(100) | **90** | **90** | **90** | **90** | **90** | **90** | **90** | **90** | 89 | **90** | 89 | 89 | 53 |
| ParcPrint(50) | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 | **50** |
| Parking(40) | 6 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 2 | 1 | 3 | 1 | **8** |
| PegSol(50) | 48 | 48 | 48 | 48 | **50** | 48 | 49 | 48 | 48 | 48 | **50** | 48 | 48 |
| PipesNT(50) | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | **21** |
| PipesT(50) | **17** | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | **17** |
| Rovers(40) | **14** | **14** | 13 | **14** | **14** | **14** | 13 | **14** | 13 | **14** | **14** | 12 | 10 |
| Satellite(36) | 9 | 10 | 9 | 9 | 10 | 9 | 9 | 9 | 9 | 9 | 10 | 9 | **16** |
| Scanlz(50) | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | **31** |
| Sokoban(50) | 48 | 48 | 48 | 48 | 48 | 48 | 48 | 48 | 48 | 48 | 48 | 48 | **50** |
| Tetris(17) | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | 9 | **10** | 9 | 8 |
| Tidybot(40) | 25 | 20 | 20 | 20 | 20 | 20 | 20 | 17 | **27** | **27** | **27** | 17 | 23 |
| TPP(30) | **9** | 8 | **9** | **9** | 8 | **9** | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| Transport(70) | **33** | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | **33** | 24 |
| VisitAll(40) | 18 | 18 | 18 | 18 | 18 | 18 | **19** | **19** | **19** | 18 | 18 | 18 | 18 |
| Woodwrk(50) | 45 | 45 | 45 | 45 | 45 | 45 | 44 | 43 | 43 | 43 | 43 | 43 | **48** |
| Zenotrvl(20) | 10 | 11 | 12 | 11 | 11 | 12 | 12 | 12 | 12 | 12 | 11 | 10 | **13** |
| Total(1607) | 968 | 954 | 955 | 959 | 965 | 967 | 963 | 960 | 964 | 959 | **973** | 936 | 962 |
| Score(40) | 20.93 | 20.78 | 20.81 | 20.93 | 21.09 | 21.17 | 21.06 | 21.11 | 21.24 | 21.00 | **21.44** | 20.04 | 19.99 |

Table 1: Coverage of SymBA* with different abstraction strategies, compared with SB and Metis.

continued. We use six different variable orderings. *lev* and *rev* use the variable ordering of the BDD and its reverse, respectively. *rnd* is a completely random ordering. *cgl* and *cgr* preserve variables interconnected in the causal graph giving preference to goal variables and breaking ties by *lev* or *rnd*, respectively. Finally, *gcl* preserves goal variables preferring those that are interconnected in the causal graph and breaking ties by *lev*. We also include the two configurations of SymBA* used in IPC'14, which use a combination of abstraction strategies in a round-robin schema. *ipc*1 uses *cgr*, *gcl* and *rev*. *ipc*2 uses the same PDB strategies plus a M&S strategy based on bisimulation with a limit of 10 000 abstract states. Finally, $\emptyset$ is a strategy that stops the algorithm instead of using any abstraction, to understand in which cases abstractions are being used. As an ablation study, we run all configurations disabling the perimeter abstractions ($\neg P$) and substituting the bidirectional search in the abstract state spaces for a standard backward search ($\neg B$).

Table 1 compares SymBA* with different abstraction strategies against the current state-of-the-art planner in symbolic search, bidirectional uniform-cost search (SB), and one state-of-the-art explicit-state search planner, METIS [Alkhazraji *et al.*, 2014]. Domains in which all planners got the same coverage are excluded from the table. We report total coverage and a final score that gives the same weight to every domain (versions of the same domain in different competitions are considered the same domain), normalizing the coverage of every planner by the number of problems in that domain. The results show that the use of abstractions can improve the results of our baseline, SB, in 25 problems from 14 different domains. However, generating abstractions implies a non-negligible overhead that affects negatively the coverage in 35 problems from 20 different domains. The conclusion is that using abstraction heuristics is beneficial in domains where the "right" abstraction is selected. Since the performance of all the strategies is close to a random selection of variables (*rnd*), all configurations are nearly tied in total coverage. The best strategy is *ipc*2, which won the optimal-track of IPC'14. With it, the standard configuration of SymBA* gets a score of 21.24; more than the baseline and METIS.

In the 936 problems solved by $\emptyset$, all the configurations behave in the same way. In the rest of the problems, abstractions are used in around 500 cases. This reflects that, by limiting the frontier size to 10 million BDD nodes, SymBA* is able to identify in which cases the blind search is going to fail and resort to abstractions. If this parameter is set to 1 million nodes, most configurations improve in domains where they are better than the baseline (e.g., *ipc*1/2 solve 18 problems in NoMystery except for $\neg B$ and *ipc*2 with $\neg B$ solves 11 instances in Satellite.) but their total coverage slightly decreases: $\emptyset$ solves 898 instances, *ipc*2 961 and *ipc*2 with $\neg B$, 962.

The ablation study shows that using perimeter abstractions is usually helpful. Regarding bidirectional search in abstract state spaces, however, the results are less clear. When a single PDB is used, both versions are closely tied though sometimes working best in different domains. However, when combining several strategies, as in the *ipc* configurations, the version disabling bidirectional search in abstract state spaces obtains better results. Our best configuration is *ipc*2 with backward search in the abstract state spaces, improving the previous state-of-the-art techniques both in coverage and score.

## 9 Conclusions

This paper addresses the question of whether heuristics can be used to further improve the results of symbolic bidirectional uniform-cost search (SB). This is a hard task, given that multiple BHS algorithms have been proposed in the past failing to outperform A* search and SB. We have introduced a new algorithm, SymBA*, that uses bidirectional A* with abstraction heuristics. SymBA* leverages the performance of SB by deferring the use of heuristics until a blind search seems unfeasible. In order to generate heuristics for both search frontiers, a bidirectional search is carried out in an abstract state space, initialized with the current frontier as a perimeter abstraction. To this end, we extended the definition of partial and perimeter abstractions to the bidirectional case. These extensions are not limited to SymBA* and they can be applied to other algorithms that use A* to explore abstract state spaces such as the hierarchical heuristic search algorithm Switchback [Larsen *et al.*, 2010; Leighton *et al.*, 2011].

Our experimental results show that abstractions can further improve the current state-of-the-art in symbolic bidirectional search, helping SymBA* to win the optimal track of the last IPC. However, finding the right abstractions in a domain-independent way is not a trivial task and there is still room for improvement in future work.

## Acknowledgments

We'd like to thank Rosa Moreno Morales for her advice and support. This work was partially supported by MICINN projects TIN2014-55637-C2-1-R and TIN2011-27652-C03-02.

## References

[Alcázar and Torralba, 2015] Vidal Alcázar and Álvaro Torralba. A reminder about the importance of computing and exploiting invariants in planning. In *Proceedings of ICAPS*, 2015.

[Alcázar *et al.*, 2014] Vidal Alcázar, Susana Fernández, and Daniel Borrajo. Analyzing the impact of partial states on duplicate detection and collision of frontiers. In *Proceedings of ICAPS*, pages 350–354, 2014.

[Alkhazraji *et al.*, 2014] Yusra Alkhazraji, Michael Katz, Robert Matmüller, Florian Pommerening, Alexander Shleyfman, and Martin Wehrle. Metis: Arming fast downward with pruning and incremental computation. In *International Planning Competition (IPC)*, pages 88–92, 2014.

[Anderson *et al.*, 2007] Kenneth Anderson, Robert Holte, and Jonathan Schaeffer. Partial pattern databases. In *Proceedings of SARA*, pages 20–34, 2007.

[Barker and Korf, 2015] Joseph Kelly Barker and Richard E. Korf. Limitations of front-to-end bidirectional heuristic search. In *Proceedings of AAAI*, pages 1086–1092, 2015.

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

[Culberson and Schaeffer, 1998] Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

[de Champeaux, 1983] Dennis de Champeaux. Bidirectional heuristic search again. *Journal of the ACM*, 30(1):22–32, 1983.

[Dillenburg and Nelson, 1994] John F. Dillenburg and Peter C. Nelson. Perimeter search. *Artificial Intelligence Journal*, 65(1):165–178, 1994.

[Edelkamp and Kissmann, 2008a] Stefan Edelkamp and Peter Kissmann. Limits and possibilities of BDDs in state space search. In *Proceedings of AAAI*, pages 1452–1453, 2008.

[Edelkamp and Kissmann, 2008b] Stefan Edelkamp and Peter Kissmann. Partial symbolic pattern databases for optimal sequential planning. In *Proceedings of KI*, pages 193–200, 2008.

[Edelkamp *et al.*, 2012] Stefan Edelkamp, Peter Kissmann, and Álvaro Torralba. Symbolic $A^*$ search with pattern databases and the merge-and-shrink abstraction. In *Proceedings of ECAI*, pages 306–311, 2012.

[Edelkamp, 2001] Stefan Edelkamp. Planning with pattern databases. In *Proceedings of ECP*, pages 13–34, 2001.

[Eyerich and Helmert, 2013] Patrick Eyerich and Malte Helmert. Stronger abstraction heuristics through perimeter search. In *Proceedings of ICAPS*, pages 303–307, 2013.

[Felner and Ofek, 2007] Ariel Felner and Nir Ofek. Combining perimeter search and pattern database abstractions. In *Proceedings of SARA*, pages 155–168, 2007.

[Felner *et al.*, 2011] Ariel Felner, Uzi Zahavi, Robert Holte, Jonathan Schaeffer, Nathan R. Sturtevant, and Zhifu Zhang. Inconsistent heuristics in theory and practice. *Artificial Intelligence Journal*, 175(9-10):1570–1603, 2011.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Helmert *et al.*, 2007] Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of ICAPS*, pages 176–183, 2007.

[Helmert *et al.*, 2014] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM*, 61(3):16:1–16:63, 2014.

[Helmert *et al.*, 2015] Malte Helmert, Gabriele Röger, and Silvan Sievers. On the expressive power of non-linear merge-and-shrink representations. In *Proceedings of ICAPS*, 2015.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research (JAIR)*, 26:191–246, 2006.

[Holte *et al.*, 2016] Robert C. Holte, Ariel Felner, Guni Sharon, and Nathan R. Sturtevant. Bidirectional search that is guaranteed to meet in the middle. In *Proceedings of AAAI*, 2016.

[Kaindl and Kainz, 1997] Hermann Kaindl and Gerhard Kainz. Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research*, 7:283–317, 1997.

[Kissmann, 2012] Peter Kissmann. *Symbolic Search in Planning and General Game Playing*. PhD thesis, Universität Bremen, 2012.

[Kwa, 1989] James B. H. Kwa. BS*: An admissible bidirectional staged heuristic search algorithm. *Artificial Intelligence Journal*, 38(1):95–109, 1989.

[Larsen *et al.*, 2010] Bradford John Larsen, Ethan Burns, Wheeler Ruml, and Robert Holte. Searching without a heuristic: Efficient use of abstraction. In *Proceedings of AAAI*, pages 114–120, 2010.

[Leighton *et al.*, 2011] Michael J. Leighton, Wheeler Ruml, and Robert C. Holte. Faster optimal and suboptimal hierarchical search. In *Proceedings of SoCS*, pages 92–99, 2011.

[McMillan, 1993] Kenneth L. McMillan. *Symbolic model checking*. Kluwer Academic publishers, 1993.

[Nilsson, 1982] Nils J. Nilsson. *Principles of Artificial Intelligence*. Springer, 1982.

[Pohl, 1969] Ira Pohl. *Bi-directional and heuristic search in path problems*. PhD thesis, Department of Computer Science, Stanford University., 1969.

[Torralba and Alcázar, 2013] Álvaro Torralba and Vidal Alcázar. Constrained symbolic search: On mutexes, BDD minimization and more. In *Proceedings of SoCS*, pages 175–183, 2013.

[Torralba *et al.*, 2013] Álvaro Torralba, Carlos Linares López, and Daniel Borrajo. Symbolic merge-and-shrink for cost-optimal planning. In *Proceedings of IJCAI*, pages 2394–2400, 2013.