Bidirectional Heuristic Search: Expanding Nodes by a Lower Bound*

Shahaf S. Shperberg[1] , **Ariel Felner**[2] , **Nathan R. Sturtevant**[2] , **Eyal Shimony**[1] and **Avi Hayoun**[1]

[1]Computer Science Department, Ben-Gurion University of the Negev, Be'er-Sheva, Israel
[2]SISE Department, Ben-Gurion University of the Negev, Be'er-Sheva, Israel
[3]Computing Science Department, University of Alberta, Edmonton, Canada
shperbsh@post.bgu.ac.il, felner@bgu.ac.il, nathanst@ualberta.ca, {shimony,hayounav}@cs.bgu.ac.il

Abstract

Recent work on bidirectional search defined a lower bound on costs of paths between pairs of nodes, and introduced a new algorithm, NBS, which is based on this bound. Building on these results, we introduce DVCBS, a new algorithm that aims to to further reduce the number of expansions. Generalizing beyond specific algorithms, we then propose a method for enhancing heuristics by propagating such lower bounds (lb-propagation) between frontiers. This lb-propagation can be used in existing algorithms, often improving their performance, as well as making them "well behaved".

1 Introduction and Background

Bidirectional heuristic search (Bi-HS) algorithms interleave two separate searches: forward from $start$, and backward from $goal$. Recently, Eckerle *et al.* [2017] defined conditions on pairs of nodes that must be expanded by Bi-HS algorithms to guarantee solution optimality. Chen *et al.* [2017] reformulated these conditions as a lower bound (lb) on costs of paths between nodes, and used this lb to introduce NBS, a non-parametric Bi-HS algorithm with an upper bound on the node expansions required to verify solution optimality.

In this paper* we present DVCBS, a new algorithm that uses the lb information differently than NBS and empirically compare them, showing that DVCBS outperforms NBS on average. In addition, we generalize beyond specific algorithms and show a connection between utilizing the lb information and two desirable properties for Bi-HS algorithms: the *well-behaved* property, where using a better heuristic never harms performance; and the *reasonable* property, that guarantees an algorithm never expands nodes with an lb greater than the known global lower bound (LB) on an optimal solution. Then, we present *lb-propagation*, a method for sharing the best lb between the two search frontiers, improving the h- and f-values in each frontier. Finally, we study a number of algorithms and show that adding lb-propagation makes some (but not all) algorithms well-behaved and reasonable, and empirically reduces the number expanded nodes in many cases.

1.1 Definitions and Notations

Given a distance metric $d(x, y)$ between nodes in an (implicitly defined) graph $G = \{V, E\}$, the aim of a shortest-path problem is to find a least-cost path between two given vertices $start$ and $goal$ in G, with $C^* = d(start, goal)$.

Most Bi-HS algorithms maintain two open lists: $Open_F$ for the forward search and $Open_B$ for the backward search.

Given a direction D (either forward or backward) We use f_D, g_D and h_D to indicate f-, g-, and h-values in direction D. W.l.o.g., the known minimal edge cost in G is denoted by ϵ, with $\epsilon = 0$ if a greater value is not known.

The heuristic h_D in direction D is *admissible* iff $h_D(u) \leq d(u, g)$ for every node $u \in G$ and is *consistent* iff $h_D(u) \leq d(u, u') + h_D(u')$ for all $u, u' \in G$. Let I_{AD} be the set of problems with admissible heuristics, and $I_{\text{CON}} \subseteq I_{\text{AD}}$ be the set of problems with consistent heuristics. A search algorithm is admissible on a set of problems I if it is guaranteed to find an optimal solution on every problem $i \in I$. Finally, a heuristic h_1 is said to *dominate* another heuristic h_2 if for every node $n \in G$, $h_1(n) \geq h_2(n)$ [Russell and Norvig, 2016].

1.2 Guaranteeing Solution Optimality

Unidirectional search algorithms must expand all nodes n with $f(n) < C^*$ in order to guarantee solution optimality [Dechter and Pearl, 1985]. Eckerle *et al.* [2017] generalized this to Bi-HS by examining pairs of nodes $\langle u, v\rangle$ with $u \in Open_F$ and $v \in Open_B$.[1]

Definition 1. *For each pair of nodes (u, v) let* $lb(u, v) = \max\{f_F(u), f_B(v), g_F(u) + g_B(v) + \epsilon\}$

In Bi-HS, a pair of nodes $\langle u, v\rangle$ is called a *must-expand pair* (MEP) if $lb(u, v) < C^*$. For each MEP only *one* of u or

*This paper summarizes the paper: "Improving Bidirectional Heuristic Search by Bound Propagation" ([Shperberg *et al.*, 2019b]) which won the best-paper award at SoCS-2019, with some additions from the paper: "Enriching non-parametric Bidirectional Search Algorithms" ([Shperberg *et al.*, 2019a]).

[1]This analysis of Eckerle *et al.* [2017] (Extended by [Chen *et al.*, 2017; Shaham *et al.*, 2018]) relies on the standard assumptions that the algorithms are deterministic, black-box, and expansion-based, and that they are admissible on I_{AD} when solving problems in I_{CON}. We therefore, assume that we are given problems from I_{CON}. However, the methods presented in this paper can be extended to algorithms that are admissible on I_{CON} but not on I_{AD} ([Shaham *et al.*, 2018; Alcázar *et al.*, 2020]).

v *must* be expanded in order to ensure that there is no path from $start$ to $goal$ passing through u and v of cost $< C^*$. In the special case of unidirectional search, algorithms expand all the nodes with $f_F < C^*$, which is equivalent to expanding the forward node of every MEP. Bi-HS algorithms may expand nodes from either side, potentially covering all the MEPs with fewer expansions.

In order to bound the minimal solution cost that can pass through each (single) node u in the open lists, we use the bound $lb(u, v)$, apply it to every node v in the opposite frontier and take the minimum of these values. Formally, for every u in $Open_D$ let $lb(u) = \min_{v \in open_{\overline{D}}}\{lb(u, v)\}$ ($\overline{D}$ is opposite to D). Then $lb(u)$ is a lower bound on the cost of any solution that passes through u. Finally, we define the global lower bound LB to be the minimal $lb(u)$ among all nodes.

lb was reformulated by Chen *et al.* [2017] as a bipartite graph (called a *Must-Expand Graph* or G_{MX}) in which for each node $u \in G$ there is a left vertex u_F and a right vertex u_B; a G_{MX} has an edge between a vertex u_F and a vertex v_B iff (u, v) is an MEP. Under this representation, the task of covering all the MEPs is equivalent to finding a vertex cover (VC) of the G_{MX}, and the minimum vertex cover (MVC) is the minimum number of expansions required to prove solution optimlaity. Chen *et al.* [2017] then used the G_{MX} representation to develop NBS, a Bi-HS algorithm that chooses an edge form the G_{MX} in every iteration, and expands both nodes. NBS terminates once a solution with cost $\leq LB$ is found. By only expanding edges of the G_{MX}, NBS is bounded by $2 \times |\text{MVC}|$ and thus expands at most twice the number of nodes required to verify solution optimality.

2 Bidirectional Search using Dynamic VC

Based on the above lower bounds, we introduce the Dynamic Vertex Cover Bidirectional Search (DVCBS) algorithm. Like NBS, DVCBS maintains LB and terminates when a solution with cost $\leq LB$ is found. However, DVCBS differs conceptually from NBS: while NBS always expands both nodes of a chosen edge (MEP), DVCBS works by maintaining a *dynamic* version of a G_{MX} (a DG_{MX}), greedily expanding a minimum vertex cover of the DG_{MX} at each step.

The structure of a DG_{MX} resembles that of a G_{MX}, with two main differences: (1) The full G_{MX} is unknown during the search. As a result, a DG_{MX} constructs left and right vertices using only nodes in $Open_F$ and $Open_B$ respectively. (2) The value of C^* is not known during the search, thus edges of a DG_{MX} are defined on pairs $\langle u, v\rangle$ such that $lb(u, v) < LB$. Since $LB \leq C^*$, all such pairs are MEPs in the relevant full G_{MX}. Note that nodes with the same g-value in the G_{MX} (and DG_{MX}) can be merged into a single *weighted vertex*, or *cluster*, where the task is to find a *minimum weighted VC*.

2.1 Choosing Nodes for Expansion

There are many ways to choose nodes for expansion given a DG_{MX} and an MVC. Every expansion deletes vertices of, and may add new vertices to the DG_{MX}, invalidating the given MVC. However, computing an MVC every time the DG_{MX} changes incurs some overhead. Thus, an efficient expansion policy should balance between expanding more nodes and maintaining relevant DG_{MX} and MVC. We experimented with various expansion policies, and found that a good balance is to expand a single cluster before re-computing the DG_{MX} and MVC. This results in a manageable number of MVC computations, while working with reasonably up-to-date information. Moreover, since all vertices in a cluster share g-values, LB may only be increased after an entire cluster is expanded. We only report experimental results for this variant.

Domain	Heuristic	Algorithm	VC: G_{MX}	Total
14 Pancake	GAP	NBS	47	147
		DVCBS	**30**	**121**
	GAP-1	NBS	5,870	5,915
		DVCBS	**4,321**	**4,344**
	GAP-2	NBS	137,295	137,719
		DVCBS	**86,292**	**87,012**
15 Puzzle	MD	NBS	12,709,517	12,748,107
		DVCBS	**11,589,837**	**11,669,720**
Grids DAO	Octile	NBS	6,561	6,677
		DVCBS	**5,158**	**5,545**
TOH4	10+2	NBS	232,509	232,509
		DVCBS	**224,233**	**224,249**
	6+6	NBS	663,136	681,995
		DVCBS	**636,375**	**664,469**

Table 1: Average node expansions across domains (using $\epsilon = 1$)

DVCBS contains several other decision points. First, there can be many possible MVCs for a given DG_{MX}. Additionally, as mentioned above, one cluster from an MVC should be chosen and expanded. Finally, the order of node expansions within a cluster may affect the number of expansions before reaching a solution when $LB = C^*$. We experimented with many variants but found the best to be as follows: select the cluster with the least amount of nodes among the clusters with minimal g_F- and g_B-values, among all MVCs. Tie breaking within a cluster orders nodes according to the order of their discovery. The results reported in Section 2.2 use this variant.

The core of DVCBS includes the following steps: (1) initialize the DG_{MX}, (2) find an MVC, (3) expand a cluster from the MVC and (4) update the DG_{MX}. Steps 2-4 repeat until either an optimal solution is found or no solution can exist.

While DVCBS outperforms NBS on average (see experiments below), DVCBS is not bounded in its worst case.

2.2 Evaluating DVCBS

In order to empirically evaluate DVCBS, we ran experiments on the following domains: **(1)** 50 **Pancake Puzzle** (with 14 pancakes) instances with the GAP heuristic [Helmert, 2010]. To get a range of heuristic strengths, we also used the GAP-n heuristics ($n = 1 \ldots 9$) where the n smallest pancakes are deleted from the heuristic computation; **(2)** 50 **12-disk, 4-peg Towers of Hanoi** (TOH4) instances with additive PDBs [Felner *et al.*, 2004]. **(3) Grid**-based pathfinding: maps from Dragon Age Origins (DAO) [Sturtevant, 2012], each with different start and goal points. **(4)** The standard 100 instances of the **15 Puzzle** problem [Korf, 1985] using Manhattan Distance. Our results are reported in Table 1 in which "VC: G_{MX}" denotes the nodes expansions before finding a VC of the G_{MX}, and "Total" denotes that overall total expansions.

As NBS has a 2× bound guarantee, any algorithm must expand at least half of those expanded by NBS, limiting possible improvements. Nevertheless, we found DVCBS required less expansions to find a VC of the G_{MX} and a solution. Finally, the expansion rates of both algorithms were similar, with very low variance. Therefore, the number of node expansions reported in Table 1 reflects the run-time accurately.

Both DVCBS and NBS are algorithms specifically designed to utilize $lb(u, v)$. We now introduce interesting theoretical properties that may be gained by using the lb information. In addition, we propose a way to incorporate this information into existing algorithms that do not include it by design.

3 Well-Behavedness and Reasonableness

If h_1 and h_2 are consistent heuristics and h_1 dominates h_2, then A* using h_1 will expand a subset of the nodes expanded when using h_2, up to tie-breaking in the last f-layer [Holte, 2010]. Holte *et al.* [2017] defined the meet-in-the-middle (MM) Bi-HS algorithm and presented an anomaly in which the above property is violated. Specifically, they presented an example in which MM_0 (an MM variant with a global zero-heuristic, h_0) expands a subset of nodes that are expanded by MM with a stronger heuristic. Barley *et al.* [2018] also refer to this anomaly, calling algorithms *well-behaved* if using a stronger heuristic does not cause any additional node expansions, and *ill-behaved* otherwise. Well-behavedness has not been formally defined in a general manner. Therefore, we introduce a general definition of the *well-behavedness* property below and show that the anomaly stems from a combination of (1) different tie-breaking, and (2) ignoring the theoretical lower-bound conditions in the expansion process.

Many heuristic search algorithms only partially specify which node to expand at each step. For example, A* may choose any node in OPEN with a minimal f-value. Thus, these algorithms specify a set of nodes from the open lists (the *allowable-set*) from which the next node must be expanded. A *tie-breaking* scheme is used to select a node from the allowable-set. Such schemes are usually implementation-specific, rather than part of the algorithm definition.

Let $\mathcal{A}_h(I, t)$ be the sequence of nodes expanded by running algorithm $\mathcal{A}$ using heuristic h on problem instance I with a tie-breaking function t, and let $S(\mathcal{A}_h(I, t))$ be a (unordered) set of nodes induced by the expansion performed by $\mathcal{A}_h(I, t)$.

Definition 2. *Let h_1, h_2 be admissible, consistent heuristics, such that h_1 dominates h_2. Algorithm $\mathcal{A}$ is said to be* well-behaved *if for every tie-breaking policy t and problem instance I, there exists a tie-breaking policy t' such that $S(\mathcal{A}_{h_1}(I, t')) \subseteq S(\mathcal{A}_{h_2}(I, t))$.*

This general definition applies to any Bi-HS algorithm. To date, only A* and GBFHS were shown to be well-behaved, and MM has been shown to be ill-behaved. Based on lb, We define conditions that determine whether algorithms are well-behaved, covering many more algorithms.

Theorem 1. *An admissible Bi-HS algorithm $\mathcal{A}$ is well-behaved if the following three sufficient (but not necessary) conditions hold:* ***(C1)*** *$\mathcal{A}$ chooses a node u for expansion* **only if** *$lb(u) = LB$;* ***(C2)*** *$\mathcal{A}$ terminates when a solution with cost $C \leq LB$ is found; and* ***(C3)*** *the allowable-set of $\mathcal{A}$ contains* **every** *node u with $lb(u) = LB$.*

Algorithm	Without lb-p		With lb-p	
	R	WB	R	WB
BHPA	×	✓	✓	✓
BS*	×	×	✓	×
fMM	×	×	✓	✓
GBFSH	✓	✓	✓	✓
NBS, DVCBS	✓	×	✓	×

Table 2: Algorithm properties summary. R columns denote reasonableness, WB columns denote well-behavedness.

While being well-behaved is an interesting property, some well-behaved algorithms that do not satisfy C1-C3, do not behave sensibly. For example, an algorithm that completely ignores heuristic values and expands nodes by their g-value is clearly well-behaved because a stronger heuristic would not affect the algorithm's behavior. However, such an algorithm might expand nodes n with $f(n) > C^*$ whose $g(n) \leq C^*$. Gilon *et al.* [2016] denoted algorithms as *reasonable* if they prune any node n with $f(n) > C$, where C an upper bound on the cost. We generalize this notion as follows:

Definition 3. *A Bi-HS algorithm is* reasonable *if for every tie-breaking policy it does not expand a node v if either $lb(v) > C^*$, or if $lb(v) = C^*$ and a solution of cost C^* was found.*

Theorem 2. *Any admissible Algorithm $\mathcal{A}$ that satisfies C1 and C2 is reasonable.*

To summarize, an algorithm that satisfies C1 and C2 is reasonable, and one that also satisfies C3 is well-behaved.

4 lb-propagation and its Effect on Algorithms

lb-propagation is a method for enhancing heuristics by utilizing lb-values. Let $h_{lb}(n) = lb(n) - g_D(n)$ denote the new heuristic for nodes in direction D. Maintaining and using h_{lb} (by propagating lb information between frontiers) instead of h in an algorithm is called *lb propagation*. Consider the following observations: **(1)** h_{lb} is a dynamic heuristic that considers information generated by the search in direction $\overline{D}$. Therefore, its value for a node may change as the search proceeds. **(2)** As $lb(n) \geq f_D(n)$ holds, $h_{lb}(n) \geq h_D(n)$ for every node in both directions. **(3)** h_{lb} maintains the consistency and admissibility properties of h.

Despite the fact that h_{lb} dominates $h(n)$, lb-propagation depends on the ability to efficiently compute the lb of nodes in OPEN. In some algorithms, the lb-propagation can be applied to a subset of OPEN, possibly enabling efficient lb computation (as in NBS). Another possibility is to use g-h buckets [Burns *et al.*, 2012]; this solution is very efficient when the number of possible g-values is relatively small.

h_{lb} changes the f-values of nodes to be their lb-value, thus, any algorithm that expand nodes and terminates based on f-values, and applies lb-propagation will now satisfy condition C1 and C2 and become reasonable. However, in order to be provably well-behaved, condition C3 is also needed.

Table 2 shows the effect of lb-propagation on several algorithms: BHPA [Pohl, 1971], BS* [Kwa, 1989], fMM [Shaham *et al.*, 2017], NBS, DVCBS and GBFHS [Barley *et al.*, 2018].

Algorithm	10-Pancake								TOH-10				Grid	
	GAP-0		GAP-1		GAP-2		GAP-3		8+2		6+4		DAO	
	h	h_{lb}	h	h_{lb}	h	h_{lb}	h	h_{lb}	h	h_{lb}	h	h_{lb}	h	h_{lb}
BPHA-Alt	**26**	**26**	674	**665**	9,484	**6,916**	50,804	**14,564**	26,435	**23,666**	96,102	**69,130**	368	**319**
BPHA-Min	25	**21**	465	**427**	6,375	**5,615**	34,497	**28,127**	33,770	**13,270**	159,079	**49,128**	413	**309**
BS*	**25**	**25**	**374**	682	**5,528**	5,585	30,687	**11,957**	**18,268**	18,351	73,434	**63,918**	**311**	496
fMM($^1/_4$)	**103**	115	5,348	**1,985**	30,858	**11,030**	82,396	**27,097**	22,660	**19,899**	65,364	**57,453**	414	**407**
MM	264	**76**	2,519	**682**	5,944	**1,684**	5,034	**2,040**	41,407	**34,307**	89,883	**76,852**	511	**501**
fMM($^3/_4$)	**64**	81	2,098	**1,111**	15,424	**6,002**	48,227	**13,263**	42,452	**36,933**	173,968	**158,290**	442	**434**

Table 3: Experimental results of average node expansions across domains

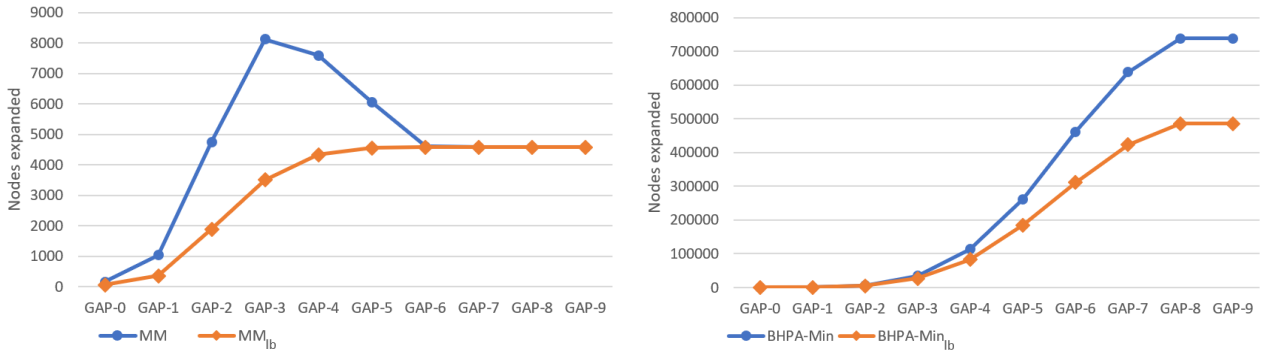

Figure 1: MM vs. MM$_{lb}$ (left) and BHPA-Min vs. BHPA-Min$_{lb}$ (right) on 10-pancake

5 Empirical Evaluation of lb-propagation

The evaluation of the lb-propagation was performed using the same settings reported in section 2.2 with the following exceptions: **(1)** 10 pancaked were used instead of 14 **(2)** 10 disk were used in TOH4 instead of 12. **(3)** the **15 Puzzle** problem domain was excluded. These changes were made since many of the algorithms evaluated in this section could not solve the original (more complex) problems in reasonable time.

Figure 1 shows average node expansions in the 10-pancake domain across all GAP heuristics. Figure 1 (left) compares MM and MM$_{lb}$, clearly showing that h_{lb} reduces the number of expansions in each of the GAP heuristics up until the heuristic effectively becomes h_0. The presence of the anomaly is demonstrated by the *hump-in-the-middle* [Barley *et al.*, 2018] in heuristics GAP-2 through GAP-6. By contrast, the hump-in-the-middle of MM$_{lb}$ is not present in when considering average expansions. Figure 1 (right) compares a variant of BHPA, denoted by BHPA-Min, with the lb-enhanced BHPA-Min. This variant selects the frontier that includes the node with the minimal f-value. Here too, the lb-propagation improves the search by reducing the number of nodes expanded. Even though BHPA-Min is well-behaved, the lb-propagation still improves the algorithm by making it reasonable.

Table 3 depicts the average number of nodes expanded across domains, with $\epsilon = 1$. h denotes the original heuristic, while h_{lb} denotes the lb-enhanced heuristic. We tested the algorithms BS*, fMM(p) using $p \in \{^1/_4, ^1/_2, ^3/_4\}$, BHPA-Min and BHPA-Alt (another BHPA variant that alternates expansions between the frontiers). We found that using lb-propagation reduces node expansions in most cases by up to a factor of 4. lb-propagation particularly excels when the heuristics are weak. In these cases, using h_{lb} always results in fewer expansions. This is also the case for GAP-4 through GAP-9, which do not appear in the table. Another interesting observation is that the hump-in-the-middle is less pronounced in all of the lb-enhanced algorithms we tested. We also experimented using $\epsilon = 0$, and similar trends were exhibited.

6 Summary and Conclusions

We have examined the lower bounds on paths costs between pairs of nodes (lb), and presented a dynamic-vertex-cover based Bi-HS algorithm, DVCBS, that uses lb by design. Moreover, we have empirically showed that DVCBS outperforms NBS on average, despite not having any upper bound guarantees. We analyzed the advantages of using the lb information by examining the anomaly present in some Bi-HS algorithms, where using a better heuristic may lead to more node expansions. Aiming to improve some of the anomalous algorithms, we defined the "well-behavedness" and "reasonableness" properties, and established sufficient conditions (C1, C2, C3) for them based on lb. Finally, we devised the lb-propagation scheme for utilizing the lb information in existing algorithms lacking this feature, that can be added to many Bi-HS algorithms, in some cases granting them these desirable properties. Empirical results show that modified algorithms exhibit better behavior, mitigating or eliminating the undesirable "hump-in-the-middle" effect.

Acknowledgements

Partially supported by ISF grant #844/17, by BSF grant #2017692, by NSF grant #1815660, by the Frankel center for CS at BGU, by CIFAR and NSERC.

References

[Alcázar *et al.*, 2020] Vidal Alcázar, Pat Riddle, and Mike Barley. A unifying view on individual bounds and heuristic inaccuracies in bidirectional search. In *AAAI*, 2020.

[Barley *et al.*, 2018] Michael W. Barley, Patricia J. Riddle, Carlos Linares López, Sean Dobson, and Ira Pohl. GBFHS: A generalized breadth-first heuristic search algorithm. In *SoCS*, pages 28–36, 2018.

[Burns *et al.*, 2012] Ethan Andrew Burns, Matthew Hatem, Michael J. Leighton, and Wheeler Ruml. Implementing fast heuristic search code. In *SoCS*, 2012.

[Chen *et al.*, 2017] Jingwei Chen, Robert C. Holte, Sandra Zilles, and Nathan R. Sturtevant. Front-to-end bidirectional heuristic search with near-optimal node expansions. In *Proceedings of IJCAI*, 2017.

[Dechter and Pearl, 1985] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 32(3):505–536, 1985.

[Eckerle *et al.*, 2017] Jurgen Eckerle, Jingwei Chen, Nathan R. Sturtevant, Sandra Zilles, and Robert C. Holte. Sufficient conditions for node expansion in bidirectional heuristic search. In *ICAPS*, 2017.

[Felner *et al.*, 2004] Ariel Felner, Richard E. Korf, and Sarit Hanan. Additive pattern database heuristics. *J. Artif. Intell. Res.*, 22:279–318, 2004.

[Gilon *et al.*, 2016] Daniel Gilon, Ariel Felner, and Roni Stern. Dynamic potential search - A new bounded suboptimal search. In *SoCS*, pages 36–44, 2016.

[Helmert, 2010] Malte Helmert. Landmark heuristics for the pancake problem. In *SoCS*, 2010.

[Holte *et al.*, 2017] Robert C. Holte, Ariel Felner, Guni Sharon, Nathan R. Sturtevant, and Jingwei Chen. MM: A bidirectional search algorithm that is guaranteed to meet in the middle. *Artif. Intell.*, 252:232–266, 2017.

[Holte, 2010] Robert C. Holte. Common misconceptions concerning heuristic search. In *SoCS*, 2010.

[Korf, 1985] Richard E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artif. Intell.*, 27(1):97–109, 1985.

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

[Pohl, 1971] Ira Pohl. Bi-directional search. *Machine intelligence*, 6:127–140, 1971.

[Russell and Norvig, 2016] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited,, 2016.

[Shaham *et al.*, 2017] Eshed Shaham, Ariel Felner, Jingwei Chen, and Nathan R. Sturtevant. The minimal set of states that must be expanded in a front-to-end bidirectional search. In *SoCS*, pages 82–90, 2017.

[Shaham *et al.*, 2018] Eshed Shaham, Ariel Felner, Nathan R. Sturtevant, and Jeffrey S. Rosenschein. Minimizing node expansions in bidirectional search with consistent heuristics. In *SoCS*, pages 81–98, 2018.

[Shperberg *et al.*, 2019a] Shahaf Shperberg, Avi Hayoun, Ariel Felner, Solomon Eyal Shimony, and Nathan R. Sturtevant. Enriching non-parametric bidirectional search algorithms. In *AAAI*, 2019.

[Shperberg *et al.*, 2019b] Shahaf S. Shperberg, Ariel Felner, Solomon Eyal Shimony, Nathan R. Sturtevant, and Avi Hayoun. Improving bidirectional heuristic search by bounds propagation. In *SOCS*, 2019.

[Sturtevant, 2012] Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(2):144–148, 2012.