Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

Improving Bidirectional Heuristic Search by Bounds Propagation

Shahaf S. Shperberg
Computer Science Dept.
Ben-Gurion University
Be'er-Sheva, Israel
shperbsh@post.bgu.ac.il

Ariel Felner
SISE Dept.
Ben-Gurion University
Be'er-Sheva, Israel
felner@bgu.ac.il

Nathan R. Sturtevant
Computing Science Dept.
University of Alberta
Edmonton, Canada
sturtevant@cs.du.edu

Solomon E. Shimony
Avi Hayoun
Computer Science Dept.
Ben-Gurion University
Be'er-Sheva, Israel
shimony@cs.bgu.ac.il

Abstract

Recent research on bidirectional search describes anomalies, or cases in which improved heuristics lead to more node expansions. Aiming to avoid such anomalies, this paper characterizes desirable properties for bidirectional search algorithms, and studies conditions for obtaining these properties. The characterization is based on a recently developed theory for bidirectional search, which has formulated conditions on pairs of nodes such that at least one node from every pair meeting these conditions must be expanded. Moreover, based on this must-expand-pairs theory, we introduce a method for enhancing heuristics by propagating lower bounds (lb-propagation) between frontiers. This lb-propagation can bestow the desirable properties on some existing algorithms (e.g., the MM family) while avoiding the above anomaly altogether. Empirical results show that lb-propagation reduces the number of node expansions in many cases.

1 Introduction

Bidirectional heuristic search (Bi-HS) algorithms interleave two separate searches: a search forward from $start$, and a search backward from $goal$. Recently, a new line of research into Bi-HS was spawned. Eckerle et al. (2017) defined three conditions on the node expansions required by Bi-HS algorithms to guarantee solution optimality. Following work reformulated these conditions as a *must-expand graph* (G_{MX}). It was shown that the *Minimum Vertex Cover* (MVC) of G_{MX} corresponds to the minimal number of expansions required to prove optimality (Chen et al. 2017). Finally, a number of algorithms were introduced. NBS (Chen et al. 2017) and DVCBS (Shperberg et al. 2019) are non-parametric G_{MX}-based Bi-HS algorithms that aim to find a vertex cover of G_{MX} quickly, but in different ways. *Fractional MM* (fMM(p)) (Shaham et al. 2017) is a parametric algorithm that generalizes the MM algorithm (Holte et al. 2017) by controlling the fraction p of the optimal path at which the forward and backward frontiers meet. Another parametric algorithm, GBFHS (Barley et al. 2018), iteratively increases the depth of the search by using a *split function* to determine how deep to search on each side at each iteration.

Holte et al. (2017) observed an *anomaly* where improving a heuristic caused the MM algorithm to expand more nodes. Aiming to generalize this anomaly beyond MM, Barley et al. (2018) defined that an algorithm is *well-behaved* if using a better heuristic will never hurt its performance; otherwise, it is *ill-behaved*. In this paper we expand this line of work on Bi-HS in several ways.

First, we study and develop desirable properties for Bi-HS algorithms by re-formalizing the *well-behaved* property and providing a definition which hs even more general than that of Barley et al. (2018). We also introduce the *reasonable* property which guarantees that an algorithm will never expand nodes if the lower-bound associated with them is greater than the current global lower bound (LB) on the optimal solution. We then introduce and prove sufficient conditions required to fulfill each property.

Second, building on the conditions of Eckerle et al. (2017), we introduce *lb-propagation*, a method for propagating the best lower-bound between the two search frontiers, thereby improving heuristics and the f-values in each frontier. lb-propagation can be used on top of any Bi-HS algorithm; it is already used implicitly in G_{MX}-based algorithms such as NBS and DVCBS. We show that lb-propagation causes the MM family to become well-behaved and reasonable, thereby avoiding the anomaly, although some algorithms, such as BS*, cannot be fixed in this way.

Third, we perform a study on a number of algorithms, characterizing those that are inherently well-behaved and reasonable, as well as whether or not lb-propagation bestows these properties on the algorithms. Finally, we show experimentally that lb-propagation reduces the number of node expansions for non-G_{MX}-based algorithms.

1.1 Definitions and Background

A shortest-path problem, P, is defined as a tuple $(G = \{V, E\}, start, goal)$ in which G is a graph and $start, goal \in V$. The aim of such problems is to find the least-cost path between $start$ and $goal$. Let $d(x, y)$ denote the shortest distance between x and y and let $C^* = d(start, goal)$. In some cases, the minimal edge-cost is known beforehand; this minimal cost is denoted by ϵ.

Most Bi-HS algorithms maintain two open lists: $Open_F$ for the forward search and $Open_B$ for the backward search. There are two types of heuristics in bidirectional search. *Front-to-front* heuristics (de Champeaux 1983; de Champeaux and Sint 1977) estimate the distance between any two nodes in the search space, while *front-to-end* heuristics (Kaindl and Kainz 1997) estimate the distance from any node and the $start$ or $goal$. Front-to-front heuristics may be more computationally expensive, and efficient data structures for front-to-frond algorithms do not exist. This paper only considers front-to-end heuristics.

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. In addition, $fmin_D$ and $gmin_D$ represent the minimal f- and g-values in that direction.

The *forward heuristic* h_F is *admissible* iff $h_F(u) \leq d(u,g)$ for every state $u \in G$ and is *consistent* iff $h_F(u) \leq d(u,u') + h_F(u')$ for all $u, u' \in G$. The *backward heuristic* h_B is defined analogously. A pair of forward and backward heuristic functions is *bi-admissible* if both heuristics are admissible. Likewise, such a pair is *bi-consistent* if both heuristics are consistent. A search algorithm is admissible if it is guaranteed to find an optimal solution whenever its heuristic is admissible. Finally, a heuristic h_1 is said to *dominate* another heuristic h_2 if and only if for every node $n \in G$, $h_1(n) \geq h_2(n)$ (Russell and Norvig 2016). We limit the discussion in this paper to admissible deterministic black-box expansion-based algorithms (called ***DXBB*** by Eckerle et al. (2017)) used with bi-admissible and bi-consistent heuristics.

1.2 Fractional MM

We use the MM family of algorithms as a case study, therefore briefly describe them next. MM is a Bi-HS algorithm that *meets in the middle* (Holte et al. 2017), i.e. it is guaranteed to never expand a node whose g-value exceeds $C^*/2$. Fractional MM ($\texttt{fMM}(p)$) is a generalization of MM that never expands a node in the forward direction whose g-value exceeds C^*/p, and never expands a node in the backward direction whose g-value exceeds $C^*/(1-p)$. For a given fraction $0 < p < 1$, $\texttt{fMM}(p)$ chooses a node for expansion according to the following priority functions:

$$pr_F(u) = \max\{g_F(u) + h_F(u), \frac{g_{F(u)}}{p} + \epsilon\}$$

$$pr_B(v) = \max\{g_B(v) + h_B(v), \frac{g_{B(v)}}{1-p} + \epsilon\}$$

A node with minimal priority in either direction is chosen for expansion.[1] fMM terminates when one of the following conditions is met:

- One of $Open_F$ or $Open_B$ is empty.
- There exists a node v in both open lists with $C = g_F(v) + g_B(v)$ s.t. either:
  - $fmin_F \geq C$;
  - $fmin_B \geq C$;
  - $gmin_F + gmin_B + \epsilon \geq C$; or
  - $\min\{\min_{u \in Open_F} pr_F(u), \min_{v \in Open_B} pr_B(v)\} \geq C$.

Note that MM is a special case of $\texttt{fMM}(p)$ with $p = 1/2$.

[1]For $p = 1$ or $p = 0$ fMM runs forward- or backward A*. Additionally, the original definition of fMM and MM did not include ϵ, which was introduced in later versions of the algorithms: MMϵ (Sharon et al. 2016) and fMMϵ (Shaham et al. 2018).

Shaham et al. (2017) showed that for every problem instance, there exists a fraction p^* such that $\texttt{fMM}(p^*)$ is optimally efficient and will expand the minimal number of nodes required to guarantee the optimality of its solution. However, p^* is not known a priori since it depends on the search-tree structure and the value of C^*.

2 The Well-Behavedness Property

If h_1 and h_2 are consistent heuristics and $h_1(s) \geq h_2(s)$ for all non-goal nodes (i.e., h_1 dominates h_2), then every node expanded by A* using h_1 will also be expanded by A* using h_2 up to tie-breaking in the last f-layer (Holte 2010). Holte et al. (2017) describe an anomaly that may occur in Bi-HS algorithms such that a similar property does not hold. An example is provided in which MM using a global zero-heuristic (denoted henceforth by h_0 and the MM variant using it by $\texttt{MM}_0$) expands a subset of nodes that are expanded by MM that uses a stronger heuristic. Barley et al. (2018) also refer to the above anomaly, calling algorithms *well-behaved* if switching to a stronger heuristic does not lead to the expansion of any additional nodes, and *ill-behaved* otherwise. Well-behavedness has not been formally defined in a general manner; Holte et al. (2017) did not formally define the anomaly and Barley et al. (2018) defined it using terms that are specific to the GBFHS algorithm. We introduce a general definition of the *well-behavedness* property below and show that the anomaly results from a combination of (1) different tie-breaking, and (2) not using the theoretical lower-bound conditions for guiding the expansion process.

Many heuristic search algorithms do not fully specify which single node to expand at any given point in the search. For example, A* may choose any node in OPEN with a minimal f-value, and fMM can choose any node in either open list with minimal priority. Instead, these algorithms specify a set of nodes from the open lists (denoted henceforth by *allowable-set*) from which the next node must be expanded. An additional *tie-breaking* scheme is used to select a single node from the allowable-set. Tie-breaking is often specific to a given implementation, and in most cases is not part of the published algorithm definition. For example, A* must expand nodes with the smallest f-value. There are many possible tie-breaking rules to decide how to break ties among nodes with the same f-value (e.g., smallest h or smallest g, generation order etc.). However, all of these tie-breaking functions are low-level details of A* implementations.

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

Definition 1. *Let h_1, h_2 be bi-admissible bi-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 is a general definition that can be used with any Bi-HS algorithm. To date, only A* and GBFHS have been proven to be well-behaved, while MM has been shown to be

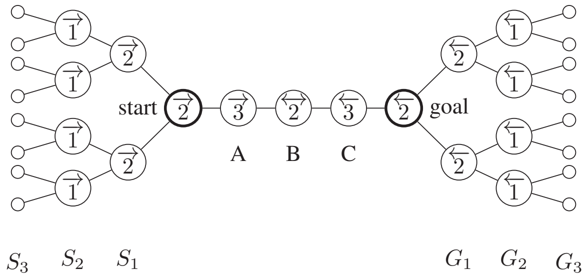

Figure 1: An example in which the anomaly manifests

ill-behaved (see below). This property has not been studied in other algorithms. We define conditions that enable the classification of algorithm as either well- or ill-behaved, covering a wider family of algorithms.

2.1 Example of the Anomaly for fMM

In order to explore algorithms that are ill-behaved, we borrow an example from Holte et al. (2017), depicted in Figure 1. In this example $\epsilon = 1$ and the values inside nodes are h-values in the direction indicated by the arrow. We henceforth denote this bi-consistent heuristic by h_{fig}. $\texttt{MM}_0$ expands nodes by their g-value. Thus, $\texttt{MM}_0$ starts by expanding $start$ and $goal$ (priority of 0), after which nodes S_1, G_1, A, and C have a priority of 1. There exists a tie-breaking policy t in which $\texttt{MM}_0$ expands A, C, and S_1 and then terminates, since it finds a solution of cost 4 and $gmin_F + gmin_B + \epsilon = 4$. In contrast, MM must expand both S_1 and G_1 after expanding $start$ and $goal$ before expanding A and C, since S_1 and G_1 have a priority of $g+h = 2g+\epsilon = 3$, while A and C have a priority of 4 ($g + h = 4$). Consequently, there exists a tie-breaking policy t for $\texttt{MM}_0$ such that for every a tie-breaking policy t' the set of nodes expanded by running MM on this instance using t' is **not** a subset of the set of nodes expanded by $\texttt{MM}_0$ using t. Thus, MM is ill-behaved.

To understand why MM is ill-behaved, consider the situation after MM expanded $start$, $goal$, and S_1. At this point, $Open_F$ contains S_2 ($g_F = 2, h_F = 1, pr_F = 5$), and A ($g_F = 1, h_F = 3, pr_F = 4$); $Open_B$ contains G_1 ($g_B = 1, h_B = 2, pr_B = 3$), and C ($g_B = 1, h_B = 3, pr_B = 4$). Thus, if the optimal solution goes through G_1, it must go through either S_2 or A. If the optimal path goes through G_1 and S_2, its cost would be at least $g_F(S_2) + g_B(G_1) + \epsilon = 4$. Similarly, if the optimal path goes through G_1 and A then its cost would be at least $f_F(A) = 4$. Hence, every path that goes through G_1 must have a cost of at least 4. The priority of G_1 ($pr_F(G_1) = 3$) doesn't reflect knowledge available in the search, which causes MM to be ill-behaved. This observation suggests that the sufficient conditions for node expansions (Eckerle et al. 2017) may be connected to whether an algorithm is well-behaved.

2.2 Guaranteeing Solution Optimality

Unidirectional search algorithms must expand all nodes n with $f(n) < C^*$ in order to guarantee the optimality of solutions (Dechter and Pearl 1985).

Eckerle et al. (2017) generalized this to Bi-HS by examining pairs of nodes $\langle u, v\rangle$ such that $u \in Open_F$ and $v \in Open_B$. Let ϵ be the minimal edge cost in G[2]. If u and v meet the following conditions, then every algorithm must expand at least one of u or v in order to ensure that there is no path from s to g passing through u and v of cost $< C^*$.

1. $f_F(u) < C^*$
2. $f_B(v) < C^*$
3. $g_F(u) + g_B(v) + \epsilon < C^*$

Definition 2. *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 $lb(u, v) < C^*$. The MEP definition is equivalent to Eckerle's conditions; for each MEP only *one* of u or v *must* be expanded. 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.

However, to address the ill-behavedness property we wish to bound the minimal solution cost that can pass through each node u in our open lists. To do so, we use the bound $lb(u, v)$ and apply it to every node v on the opposite frontier and take the minimum among these values. Formally, for every node u in $Open_D$ let

$$lb(u) = \min_{v \in open_{\overline{D}}} \{lb(u, v)\}$$

where $\overline{D}$ denotes the opposite direction from 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. This is identical to the minimal $lb(u, v)$ among all pairs. LB was used in the high-level pseudocode (described below) of the NBS and DVCBS algorithms. Note that the search begins with $LB = lb(start, goal)$, after which LB increases iteratively until $LB = C^*$. We can now use these definitions to show whether a family of algorithms is well-behaved.

2.3 Conditions for Being Well-Behaved

We first introduce three sufficient conditions for an admissible Bi-HS algorithm $\mathcal{A}$ to be well-behaved:

Condition C1: Algorithm $\mathcal{A}$ chooses a node u for expansion **only if** $lb(u) = LB$.

Condition C2: Algorithm $\mathcal{A}$ terminates as soon as a solution with cost $C \leq LB$ is found.

Condition C3: The allowable-set of algorithm $\mathcal{A}$ contains **every** node u with $lb(u) = LB$.

Theorem 1. *An admissible Bi-HS algorithm $\mathcal{A}$ that satisfies conditions C1, C2, and C3 is well-behaved.*

Proof. Let h_1, h_2 be bi-admissible bi-consistent heuristics, s.t. h_1 dominates h_2, and let t_2 be an arbitrary tie-breaking

[2]Strictly speaking the ϵ term was added by Shaham et al. (2018) as a generalization of the inequalities, since $\epsilon = 0$ is always a lower-bound to edge cost.

policy. We will show that there exists a tie-breaking policy t_1 s.t. $S_1 = S(\mathcal{A}_{h_1}(I, t_1)) \subseteq S(\mathcal{A}_{h_2}(I, t_2)) = S_2$.

To do this, we examine the execution of $\mathcal{A}_{h_1}(I, t_1 = t_2)$ and show how to modify t_1 to make $S_1 \subseteq S_2$. Let n be the first node expanded in the trace of the execution s.t. $n \notin S_2$ (if no such node exists then $S_1 \subseteq S_2$ and we are done). In addition, let D be the direction in which n was expanded. We now have two cases:

Case 1: There exists a node n' in one of the frontiers s.t. $lb(n') = lb(n) = LB$ and $n' \in S_2$. given C3, we can modify t_1 to choose n' instead of n.

Case 2: All of the nodes n' in the frontiers with $lb(n') = lb(n) = LB$ are not in S_2. We will show that this case is not possible. Note that since n was chosen for expansion, all other nodes m in the frontiers have $lb(m) \geq lb(n) = LB$. Let v be a node in $Open_{\overline{D}}$ s.t. $lb(n) = lb(n, v)$. Since $lb(v) \leq lb(n, v)$ and $lb(v) \geq lb(n)$ then $lb(n) = lb(v)$, and therefore $v \notin S_2$. Since both n and v are in OPEN using h_1 when n is chosen for expansion, and because n is the first to not be in S_2, then all other nodes that were expanded by h_1 were expanded by h_2. Thus, at some point in $\mathcal{A}_{h_2}(I, t_2)$, both n and v are in OPEN with a g-value less than or equal to that found in $\mathcal{A}_{h_1}(I, t_1)$. We can therefore refer to $lb(n, v)$ in $\mathcal{A}_{h_2}(I, t_2)$. Henceforth, $lb_i(n, v)$ refers to the value $lb(n, v)$ in $\mathcal{A}_{h_i}(I, t_i)$. Since h_1 dominates h_2, and since the g-values cannot be larger when using h_2, $lb_2(n, v) \leq lb_1(n, v)$. We proceed by examining the possible values of $lb_1(n, v)$, and show that any value leads to a contradiction. There are three possible values of $lb_1(n, v)$ to consider:

(i) If $lb_1(n, v) < C^*$, then $lb_2(n, v) < C^*$ as well. Therefore, $\langle n, v \rangle$ is an `MEP` and any admissible algorithm must expand one of them. The fact that $n, v \notin S_2$ contradicts the admissibility of $\mathcal{A}$.

(ii) If $lb_1(n, v) = C^*$, then since $\mathcal{A}_{h_1}(I, t_1)$ expanded n, it did not yet find any solution of cost C^*. Since LB is a lower-bound on the optimal solution, a solution with cost C^* must pass through some node m with $lb(m) = lb(n) = C^*$ that was not yet expanded. Under the assumption of case 2, there are no nodes m with $lb(m) = lb(n) = LB$ in one of the frontiers that is also in S_2. Therefore, $\mathcal{A}_{h_2}(I, t_2)$ will never find a solution of cost C^*. This is a contradiction to the assumption that $\mathcal{A}$ is admissible.

(iii) If $lb_1(n, v) > C^*$, then since $\mathcal{A}_{h_1}(I, t_1)$ expanded n and $\mathcal{A}$ satisfies C2, it did not find any solution of cost C^*. Additionally, when n was chosen for expansion, the lb between every pair of nodes in the open lists is greater than C^*. Thus, a solution of cost C^* does not exist by contradiction to the definition of C^*. □

These conditions are sufficient, but not necessary. In the next section we explore another desirable property.

3 The Reasonableness Property

While being well-behaved is an interesting property, some well-behaved algorithms do not behave sensibly. For example, an algorithm that completely ignores heuristic values and expands nodes according to their g-value is clearly well-behaved because a stronger heuristic will not change the behavior of the algorithm. However, such an algorithm might expand nodes n with $f(n) > C^*$ whose $g(n) \leq C^*$. Gilon, Felner, and Stern (2016) denoted algorithms as *reasonable* if they have a best-first structure (i.e. an open list and an expansion rule), and 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^* has already been found.*

Note that since $f(n) \leq lb(n)$ and $C^* \leq C$, the redefinition of the reasonable property is tighter than the original definition of Gilon, Felner, and Stern (2016).

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

Proof. Let $\mathcal{A}$ be an algorithm that always expands a node u with $lb(u) = LB$ and terminates as soon as a solution with a cost $c \leq LB$ is found. Assume by contradiction that $lb(u) > C^*$. Since $lb(u)$ is minimal ($lb(u) = LB$) then every solution that passes through every node in the open lists has a cost $> C^*$. Since C2 dictates that $\mathcal{A}$ terminates when a solution with a cost $c = LB$ is found, no solution with cost C^* could have been found. Therefore, there is no possible solution with a cost of C^*, by contradiction to the definition of C^*. □

To summarize both theorems, an algorithm that satisfies conditions C1 and C2 is reasonable, and one that also satisfies C3 is well-behaved. In both cases, the conditions are *sufficient* but not *necessary*.

4 Improving Heuristics by lb-propagation

We next introduce several methods that improve the heuristic value of a node by utilizing information gathered during the search in both frontiers. The strongest method which propagates lb-values causes some ill-behaved algorithms to become well-behaved (e.g., the `MM` family). In addition, algorithms that satisfy conditions C1 and C2 with respect to f instead of lb which use this method become reasonable. Note that this improvement is achieved by modifying only the heuristic, without any other changes to the algorithms.

4.1 Propagating g- and f-values

A simple observation on the nature of bidirectional search yields that the minimum $g_{\overline{D}}$-value with the addition of ϵ is an admissible heuristic for any node in $Open_D$. Furthermore, we can propagate the minimal f-value from the opposite frontier because it is a lower bound on any possible solution. Formally, let $gmin_D = \min_{v \in open_D} \{g(v)\}$ and let $fmin_{\overline{D}} = \min_{v \in open_{\overline{D}}} \{f_{\overline{D}}(v)\}$. We can improve the heuristic of node n in direction D to be:

$$h'_D(n) = \max\{h_D(n), gmin_{\overline{D}} + \epsilon, fmin_{\overline{D}} - g_D(n)\}$$

h' clearly dominates h and is easy to implement. One only needs to keep track of $gmin_D$ and $fmin_D$ for both directions. Nevertheless, h' does not solve the anomaly; `MM` using

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

Table 1: Algorithm properties summary. R columns denote reasonableness, WB columns denote well-behavedness.

h' on Figure 1 behaves identically to `MM` using the original heuristic, as described in Section 2.1

4.2 lb-propagation heuristic

The next heuristic exploits knowledge from lb-values. Let $h_{lb}(n) = lb(n) - g_D(n)$ denote the new heuristic function for nodes in direction D. Consider the following key observations: **(1)** h_{lb} is a dynamic heuristic that takes into account information generated by the search in the opposite direction. Therefore, its value for a node may change as the search proceeds. **(2)** Since $lb(n) \geq f_D(n)$, $h_{lb}(n) \geq h_D(n)$ for every node in both directions. **(3)** h_{lb} maintains the bi-consistency and bi-admissibility properties of h.

The heuristic h_{lb} dominates h' because h' looks at the global values of $gmin_{\overline{D}}$ and $fmin_{\overline{D}}$, while h_{lb} considers each pair of nodes in isolation. Despite the fact that h_{lb} dominates h', using lb-propagation depends on the ability to efficiently compute the lb of nodes in OPEN. This task is certainly more difficult than applying the other propagation, which simply requires maintaining the minimal f- and g-values in each direction. In some algorithms the lb-propagation can be applied to a limited subset of OPEN, possibly enabling an efficient implementation (similar to `NBS`). In other cases, the lb of every node is required. This leads to a potentially less a efficient implementation, using g-h buckets (Burns et al. 2012); this solution would work if the number of possible g-values (and therefore h-values) is relatively small, which is the case in many common domains.

An important property of h_{lb} is that it changes the f-values of nodes to be their lb-value, and therefore makes some existing algorithms well-behaved and reasonable as we show in the next section.

5 Classification of Existing Algorithms

As mentioned, lb-propagation makes the f-values of nodes identical to their lb-values. Therefore, any algorithm which chooses to expand nodes based on f-values and applies lb-propagation will now satisfy condition C1. However, in order to be provably reasonable it should also satisfy condition C2, and to be well-behaved, condition C3 is also needed. In this section, we review several Bi-HS algorithms and analyze how lb-propagation affects them. For any algorithm $\mathcal{A}$ we henceforth denote by $\mathcal{A}_{lb}$ a version of $\mathcal{A}$ that applies lb-propagation. Table 1 summarizes the results of this section, for algorithms with and without lb-propagation (lb-p).

5.1 BHPA

We begin with BHPA (Pohl 1971), a simple algorithm that first selects a direction and chooses to expand a node with minimal f-value in that direction. BHPA terminates when the minimal f-value is *greater than or equal to* C^*.

Lemma 3. *BHPA is well-behaved.*

Proof. Let I be a problem instance, h_1 and h_2 be heuristics that are bi-admissible and bi-consistent on I s.t. h_1 dominates h_2, and let t_2 be a tie-breaking policy. Let S_2 denote $S(BHPA_{h_2}(I, t_2))$ and let S_1 denote $S(BHPA_{h_1}(I, t_1 = t_2))$. Let u be the first node expanded in the trace of $BHPA_{h_1}(I, t_1)$ s.t. $u \notin S_2$. If u does not exist, we are done. Otherwise, we want to fix t_1. If there exists a node $n \in S_2$ that has a minimal f-value in either direction of $BHPA_{h_1}(I, t_1)$ when u was selected for expansion, we can alter t_1 to select n instead of u. Otherwise, there exists a node u' in the opposite direction of u with minimal f-value, and we could modify t_1 to select u' instead of u for expansion. We know that for every node v, ${f_D}^{h_1}(v) \geq {f_D}^{h_2}(v)$, thus ${f_D}^{h_1}(u) \geq {f_D}^{h_2}(u)$ and ${f_D}^{h_1}(u') \geq {f_D}^{h_2}(u')$. Therefore, if ${f_D}^{h_1}(u) < C^*$ and ${f_D}^{h_1}(u') < C^*$, we know that ${f_D}^{h_2}(u) < C^*$ and ${f_D}^{h_2}(u') < C^*$, hence S_2 must contain either u or u', so that $BHPA_{h_2}(I, t_2)$ could terminate by contradiction to the fact that $u, u' \notin S_2$. Otherwise, ${f_D}^{h_1}(u) = C^*$ or ${f_D}^{h_1}(u') = C^*$. In this case, since $BHPA_{h_1}(I, t_1)$ is admissible and must find an optimal solution, there must be some other node $v \in S_2$ in the open lists when u was chosen for expansion s.t. ${f_D}^{h_1}(v) = C^*$ by contradiction to the case assumption. ☐

Lemma 4. *BHPA is unreasonable.*

Proof. Consider the problem instance I in Figure 1 assuming that $h_F(S_3) = h_B(G_3) = 0$. Since for all $i \in \{1, 2, 3\}$, $f_F(S_i) = f_B(G_i) = 3$, while $f_F(A) = f_B(C) = 4$, running $BHPA_{h_{fig}}$ on I with any tie-breaking policy must expand either $\{S_1, S_2, S_3\}$, $\{G_1, G_2, G_3\}$, or both, before being able to expand A or C. Furthermore, there exists a tie-breaking in which $BHPA_{h_{fig}}$ expands $start$ and $goal$ followed by $\{S_1, S_2, S_3\}$. Since $lb(S_3) = g_F(S_3) + g_B(C) + \epsilon = 5 > C^* = 4$, $BHPA$ is unreasonable. ☐

Lemma 5. *BHPA$_{lb}$ is reasonable and well-behaved.*

Proof. Since after the propagation the f-value of a node equals its lb, BHPA$_{lb}$ always expands nodes with minimal lb (C1). In addition, BHPA$_{lb}$ terminate as soon as a solution with a cost $C \leq fmin_D = LB$ is found (C2). Finally, the allowable-set of BHPA$_{lb}$ contains all nodes with minimal lb since they all have the same f-value (C3). ☐

5.2 `BS*`

`BS*` (Kwa 1989) expands a node with a minimal f-value from the smallest open-list (Pohl's cardinality criterion (Pohl 1971)) and terminates when the minimal f-value is *greater than or equal to* C^*. In addition, `BS*` trims nodes from the open lists if their f-value is *greater than or equal to* costs of potential solutions that were already found.

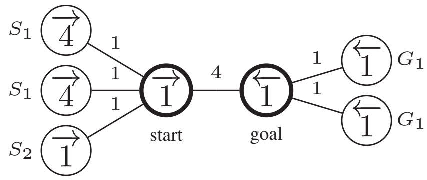

Figure 2: $\texttt{BS}^*{}_{lb}$ is not well-behaved

Lemma 6. $\texttt{BS}^*$ *and* $\texttt{BS}^*{}_{lb}$ *are ill-behaved.*

Proof. Consider problem instance I in Figure 2. Using the heuristic values written inside the nodes, $\texttt{BS}^*$ starts by expanding $start$ and $goal$ in an unspecified order. Nodes G_1 and S_2 get f-values of 2, and nodes S_1 an f-value of 5. Since $\texttt{BS}^*$ has already found a solution of cost 4, nodes S_1 are trimmed from $Open_F$. At this point $Open_F$ contains only S_2, while $Open_B$ contains two nodes (G_1). Thus, $\texttt{BS}^*$ is forced to choose S_2 for expansion before terminating.

Next, consider $\texttt{BS}^*$ using h_0. The beginning of the search is similar: $start$ and $goal$ in an unspecified order. Then all nodes of $Open_F$ get an f-value of 1. Since no node is trimmed, $Open_F$ contains 3 nodes, while the $Open_B$ contains 2 nodes. Thus, $\texttt{BS}^*$ expands the G_1 nodes before terminating, without expanding S_2.

While applying the propagation changes the f-value of nodes, the behaviour of $\texttt{BS}^*{}_{lb}$ is identical to $\texttt{BS}^*$ on this example. Thus, both $\texttt{BS}^*$ and $\texttt{BS}^*{}_{lb}$ are ill-behaved. We note that C3 is violated here because the allowable-set of $\texttt{BS}^*{}_{lb}$ is forced to contain only one open list. □

Lemma 7. $\texttt{BS}^*$ *is unreasonable.*

Proof. The proof is similar to that of Lemma 4. Consider the problem instance I in Figure 1 assuming that $h_F(S_3) = h_B(G_3) = 0$. Similar to the proof of Lemma 4, $\texttt{BS}^*$ will have to expand $start$,$goal$, $\{S_1, S_2, S_3\}$ and $\{G_1, G_2, G_3\}$ before expanding A or C. Since $lb(S_3) = g_F(S_3)+g_B(C)+\epsilon = 5 > C^* = 4$, $\texttt{BS}^*$ is unreasonable. □

Lemma 8. $\texttt{BS}^*{}_{lb}$ *is reasonable.*

Proof. Since after the propagation the f-value of a node equals its lb, $\texttt{BS}^*{}_{lb}$ always expands nodes with minimal lb (C1). Finally, $\texttt{BS}^*{}_{lb}$ terminates as soon as a solution with a cost $c \leq fmin_D = LB$ is found (C2). Therefore, both conditions are satisfied and $\texttt{BS}^*{}_{lb}$ is reasonable. □

5.3 fMM

We have already shown that fMM is ill-behaved in Section 2.1. We now show that fMM is also unreasonable.

Lemma 9. fMM *is unreasonable.*

Proof. Consider $\texttt{fMM}(1/4)$ applied to the problem instance in Figure 3. After expanding $start$ and $goal$, a solution of cost 11 is discovered, and $LB = lb(S_2, G_2) = g_F(S_2) + g_B(G_2) + \epsilon = 12$. Since $pr(G_2) = \max\{f_B(G_2), \frac{4}{3}g_B(G_2)\} = 10$, G_2 will be expanded before termination, even though $12 = LB > C^* = 11$. □

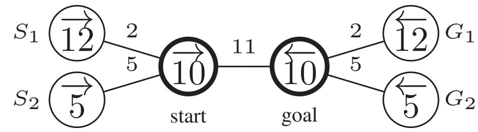

Figure 3: fMM is not reasonable

Lemma 10. $\texttt{fMM}_{lb}$ *always expands a node u with $lb(u) = LB$, hence it is reasonable.*

Proof. Assume by contradiction that $\texttt{fMM}_{lb}$ chose a node u in direction D for expansion s.t. $lb(u') \neq LB$. Therefore, there exists a pair of nodes (u', v') s.t. u' is in the $Open_D$, v' is in $Open_{\overline{D}}$ and $lb(u') = lb(v') = lb(u', v') = LB < lb(u)$. Using the lb-propagation, we know that $f_D(u') = lb(u') = lb(u', v')$, and $f_{\overline{D}}(v') = lb(v') = lb(u', v')$. Therefore, $f_D(u) = lb(u) > f_D(u')$. Likewise, $f_D(u) = lb(u) > f_{\overline{D}}(v')$.

$$\begin{aligned} f_D(u) &> f_{\overline{D}}(v') = f_D(u') = lb(u') = lb(u', v') \\ &\geq g_D(u') + g_{\overline{D}}(v') + \epsilon \end{aligned}$$

Since u was chosen for expansion, we know that $pr_D(u) \leq pr_D(u')$ and $pr_D(u) \leq pr_{\overline{D}}(v')$. Thus,

$$\max(f_D(u), \frac{g_D(u)}{p} + \epsilon) \leq \max(f_D(u'), \frac{g_D(u')}{p} + \epsilon)$$

and

$$\max(f_D(u), \frac{g_D(u)}{p} + \epsilon) \leq \max(f_{\overline{D}}(v'), \frac{g_{\overline{D}}(v')}{1-p} + \epsilon)$$

Since $f_D(u) = lb(u) > lb(u') = f_D(u') = lb(v') = f_{\overline{D}}(v')$,

$$f_D(u) \leq \frac{g_D(u')}{p} + \epsilon \tag{1}$$

$$f_D(u) \leq \frac{g_D(v')}{1-p} + \epsilon \tag{2}$$

By summing inequality (1) multiplied by p with inequality (2) multiplied by $1 - p$, we get:

$$f_D(u) = pf_D(u) + (1-p)f_D(u) \leq g_D(u') + g_{\overline{D}}(v') + \epsilon$$

In contradiction to: $f_D(u) > g_D(u')+g_{\overline{D}}(v')+\epsilon$ above. □

Lemma 11. $\texttt{fMM}_{lb}$ *(with lb-propagation) is well-behaved.*

Proof. Here we cannot use Theorem 1 directly since the allowable-set of $\texttt{fMM}_{lb}$ does not include every node u with $lb(u) = LB$, as some of these nodes might have g-values that raise their priority. Nonetheless, we show that $\texttt{fMM}_{lb}$ using lb-propagation is in fact well-behaved, using a slight modification to Theorem 1. In Lemma 10 we showed that $\texttt{fMM}_{lb}$ only expands nodes with minimal lb. In addition, $\texttt{fMM}_{lb}$ terminates when the lowest f-value is $\geq C^*$. Since the f-value of nodes after applying lb-propagation is equal their lb-value, $\texttt{fMM}_{lb}$ stops when $LB \geq C^*$. Thus, C1 and C3 are satisfied. However, C2 is violated by the priority mechanism of fMM, since nodes with the same lb-value might have different priorities due to their g-values and direction. For example, if there is only one node in $Open_F$ with $f_F = 3$,$g_F = 1$, and only one node in $Open_B$ with $f_B = 3$,$g_B = 2$, fMM will give lower priority to the node in $Open_F$, based on his g-value (priority of 3 versus a priority of 5). Nonetheless, we will show that $\texttt{fMM}_{lb}$ is still well-behaved. The problem arises since the first case of of the proof of Theorem 1 reduces to nodes n' with the $lb(n') = lb(n)$ and $pr(n') = pr(n)$. Therefore, nodes n' with $lb(n') = lb(n)$ and $pr(n') \neq pr(n)$ are part of the second case of the proof, which does not cover them. In that case, we considered some v s.t. $lb(n) = lb(v) = lb(n, v)$. Following that, it was clear that $v \notin S_2$. Nonetheless, in our case, v could have been in S_2 if it had a different priority than n when $\texttt{fMM}_{lb}$ was running using h_1. Since $\texttt{fMM}_{lb}$ expands nodes with minimal priority, $pr(v) > pr(n)$. The priority of v could have been determined by one of the following options:

Case 1: $f_{\overline{D}}(v) = lb(v) = pr(v)$. However, $lb(u) \leq pr(u)$ and $lb(v) = lb(u)$. Therefore, $pr(v) \leq pr(u)$, by contradiction to the assumption that $pr(v) > pr(u)$.

Case 2: $\frac{g_{\overline{D}}(v)}{1-p} = pr(v)$. Since $f_{\overline{D}2}(v) \leq f_{\overline{D}1}(v)$ and $f_{\overline{D}}(v) \leq \frac{g_{\overline{D}}(v)}{1-p} = pr(v)$, we know that the $pr(v)$ using h_2 is less or equal than $pr(v)$ using h_1. In addition, the priority of node n (and any of its ancestors) when running using h_1 must be strictly less than $pr(v)$, by contradiction to the fact that v was already chosen for expansion.

Therefore, we can conclude that v is still not in S_2 and the proof of Theorem 1 is generalized to fMM as well. □

5.4 NBS and DVCBS

NBS (Chen et al. 2017) and DVCBS (Shperberg et al. 2019) are two prominent Bi-HS algorithms that choose nodes for expansion with minimal lb. At any point in the search, NBS chooses a pair of nodes with minimal lb and expands them both. DVCBS expands nodes from a subset of those with minimal lb, determined by maintaining a dynamic version of the G_{MX} (denoted by DG_{MX}) and finding its MVC. Both algorithms terminate as soon as a solution with a cost $c \leq LB$ is found. Since the expansion policy and termination condition of both algorithm already consider LB, their properties remain unaffected by lb-propagation.

Clearly, NBS and DVCBS satisfy conditions C1, and C2 and are therefore reasonable (up to a single additional expansion). However as previously mentioned, the allowable-set of DVCBS includes only nodes that make up the MVC of DG_{MX}, violating C3. In addition, once NBS has chosen a pair (u, v) for expansion, it is committed to expanding both nodes. Therefore, after expanding u, any node u' in the same direction of u s.t. $lb(u') = lb(u) = lb(v) = LB$ is not in the allowable set of NBS until after expanding v, in violation of condition C3.

Lemma 12. NBS *is ill-behaved.*

Proof. Consider the problem instance of Figure 4.[3] NBS using h_0 will start by expanding $start$ and $goal$. Afterwards, X and A are added to $Open_F$, and C is added to $Open_B$. Since $g_F(x) = f_F(x) = 12$, $lb(X, C) = 12$, while $lb(A, C) = 9$. Therefore, the pair (A, C) is chosen for expansion, after which a path of length 20 has been discovered. Since $lb(X, B) \geq 20$ and $lb(B, B) \geq 20$, NBS terminates after expanding $start, A, C$, and $goal$.

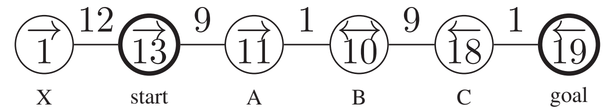

Figure 4: NBS is not well-behaved

NBS using the heuristic in the nodes starts by expanding $start$ and $goal$. Next, X and A are added to $Open_F$, and C is added to $Open_B$. Since $lb(A, C) = 20$ (due to $f_F(A)$) and $lb(X, C) = 19$ (due to $f_B(C)$). Therefore, NBS will expand the pair (X, C), despite the fact that X was not expanded using a weaker heuristic. □

Lemma 13. DVCBS *is ill-behaved.*

Proof. Consider the problem instance of Figure 2. DVCBS using the heuristic in the nodes starts by expanding either $start$ or $goal$ since both of them are MVCs of DG_{MX}. If $goal$ was chosen, $start$ must be expanded, since it becomes the only MVC, followed by S_2 for a similar reason, after which DVCBS terminates since $LB = 5 > 4$ (the path that was discovered from $start$ to $goal$). Likewise, if $start$ was chosen for expansion, DVCBS will expand either S_1 and terminate, or $goal$ followed by S_1. In both cases the G_1 nodes are never expanded. However, DVCBS using h_0 must expand $start$ and $goal$ in an unspecified order, followed by G_1. □

5.5 GBFSH

GBFHS is an algorithm that iteratively increases the depth of the search ($fLim$). At each depth, a pre-defined *split function* (parameter of the algorithm) is used that determines how deep to search on each side at each iteration by splitting $fLim$ to $gLim_F$ and $gLim_B$ s.t. $fLim = gLim_F + gLim_B + \epsilon - 1$. At every iteration, GBFHS considers nodes for expansion in direction D with $f \leq fLim$ and $g < gLim_D$. GBFHS terminates when as soon as a solution with a cost equals to $fLim$ is found.

GBFHS was proved to be well-behaved.[4] We show that GBFHS is reasonable by showing that it expands only nodes with minimal lb, even without applying lb-propagation.

Lemma 14. GBFHS *is reasonable.*

Proof. Since GBFHS considers nodes for expansion in direction D with $f \leq fLim$ and $g < gLim_D$, a node u that is chosen for expansion will have $lb(u) \leq \max\{gLim_F + gLim_B + \epsilon - 1, fLim\} = flim$ [5], and since the $flim$ is increased only after there are no nodes left for expansion,

[3]This example is due to Robert Holte and Sandra Zilles

[4]Even though well-behavedness was not defined in a general manner when GBFHS was created, the proof of Barley et al. (2018) is still applicable to the new definition with slight modifications.

[5]Barley et al. (2018) implicitly assume that ϵ is an integer ≥ 1.

$lb(u) = flim = LB$. Ergo, GBFHS expands nodes with minimal lb. In addition, GBFHS terminates as soon as a solution of cost $flim = LB$ is found. Thus, C1 and C2 are satisfied and GBFHS is reasonable. □

6 Experimental results

We ran experiments on three domains: **(1)** 50 **10-Pancake Puzzle** instances with the GAP heuristic (Helmert 2010). To get a range of heuristic strengths, we also used the GAP-n heuristics (for $n = 1 \dots 9$) where the n smallest pancakes are deleted from the heuristic computation; **(2)** 50 instances of the 10-disk **4-peg Towers of Hanoi** (TOH4) problem with (8+2) and (6+4) additive PDBs (Felner, Korf, and Hanan 2004). **(3) Grid**-based pathfinding: 65 maps from Dragon Age Origins (DAO) (Sturtevant 2012), each with different start and goal points (a total of 1,680 instances).

Figure 5 shows the average number of nodes expanded by MM and by MM_{lb} in the 10-pancake domain across all GAP heuristics. Clearly, adding the lb-propagation significantly reduces the number of nodes expanded. Using h_{lb} seems to reduce the number of node expansions for each of the GAP heuristics up until GAP-7, in which the heuristic effectively becomes h_0. In addition, this figure clearly demonstrates the anomaly of MM; the average number of nodes expanded by MM using heuristics GAP-2 through GAP-6 is greater than the number of nodes expanded by using heuristics GAP-7 through GAP-9 (notice the "hump-in-the-middle" (Barley et al. 2018)). By contrast, the hump-in-the-middle of MM_{lb} is much smaller, and in fact not visible when considering the average number of expansions. However, there were still some individual problem instances in which MM_{lb} expanded fewer nodes using a weaker heuristic. This is consistent with Theorem 1, since we are using a predetermined tie-breaking policy and not the best possible tie-breaking policy for every instance. Interestingly, MM_{lb} using $\epsilon = 0$ demonstrates no hump-in-the-middle, even when considering individual problem instances.

Similarly, Figure 6 shows the average number of nodes expanded in the 10-pancake domain across all GAP heuristics, with $\epsilon = 1$ by a variant of BHPA denoted by BHPA-Min. BHPA-Min 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, and demonstrates no hump-in-the-middle in the average case, the lb-propagation still improves the algorithm by making it reasonable. This improvement is more evident with GAP-8 and GAP-9; Despite these GAP heuristics behaving like h_0 in the 10-pancake domain, lb-propagation incorporates $gmin_F + gmin_B + \epsilon$ into the f-values of nodes in BHPA-Min, exposing an additional termination condition, and allowing the search process to halt sooner.

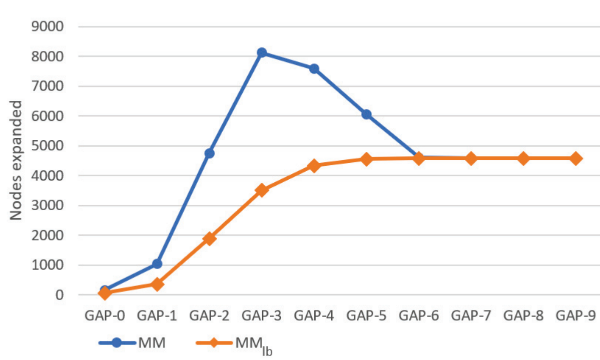

Figure 5: MM vs. MM_{lb} on 10-pancake

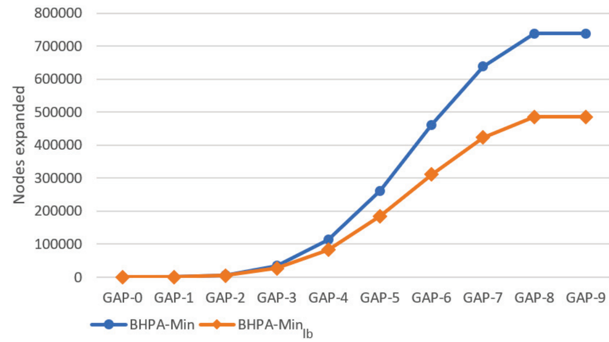

Figure 6: BHPA-Min vs. BHPA-Min$_{lb}$ on 10-pancake

The average number of node expanded across domains, using $\epsilon = 1$, appear in Table 2. There is one row for each algorithm, and one column for each of the domains and their heuristics; h denotes the original heuristic, while h_{lb} denotes the heuristic enhanced by lb-propagation. The algorithms we have tested are BS*, fMM(p) using $p \in \{1/4, 1/2, 3/4\}$, BHPA-Min and BHPA-Alt, another variant of BHPA that alternates between the frontiers between expansions. The results show that using the lb-propagation reduces the number of node expansions in most cases by up to a factor of 4. The lb-propagation particularly excels when the heuristics are weak. In these cases using h_{lb} always results in fewer node 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 tested algorithms. BS* seems to be the least affected by the propagation among all algorithms. We posit that the reason for this is that BS* is highly dependent on the search process; since BS* selects a node for expansion from the direction with the smallest open list, minor increments in heuristic values might cause nodes to be trimmed away, possibly changing the size balance of the two frontiers. BS* also assumes that the heuristic is consistent, which other algorithms do not. We have also experimented using $\epsilon = 0$; the results are similar to the reported $\epsilon = 1$ results.

Naturally, maintaining and using lb for each node incurs overheads. In the domains we used, g-h-bucketing requires negligible time and space. However, we did not focus on code optimization, and used naive data-structures. Thus, run times are not reported here. Improving efficiency with a bucketing scheme or adapting the data structures used for efficient lb computations by NBS is reserved for future work.

	10-Pancake								TOH-10				Grid	
Algorithm	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 2: Experimental results of average node expansions across domains

7 Discussion

We have examined the source of the anomaly exhibited by some Bi-HS algorithms, where using a better heuristic causes the algorithm to expand more nodes. Aiming to improve some algorithms in which the anomaly manifests, the properties of "well-behavedness" and "reasonableness" were defined, and sufficient conditions (C1, C2, C3) for these properties were established. These properties provide insights that lead to the lower-bound propagation scheme (lb-propogation) that can be added to many existing Bi-HS algorithms, in some cases bestowing upon them these desirable properties. Empirical results show that modified algorithms exhibit better behavior, alleviating or even eliminating the undesirable "hump-in-the-middle" effect seen when an algorithm is run with heuristics of varying quality.

The well-behavedness property as defined in this paper ensures that there exists a tie-breaking policy for which the anomaly would not occur. However, the desired tie-breaking policy is not specified. It is a non-trivial issue, left for future research, to define conditions that guarantee a stronger well-behavedness property of an algorithm, such that a dominating heuristic would *never* cause more nodes to be expanded than the weaker heuristic using the **same** tie-breaking policy. Another interesting research direction is to re-examine the three well-behavedness conditions with respect to heuristics that are strictly dominating, i.e., $h_1 > h_2$.

8 Acknowledgements

This work was supported by Israel Science Foundation (ISF) grant #844/17 to Ariel Felner and Eyal Shimony, by BSF grant #2017692, by NSF grant #1815660 and by the Frankel center for CS at BGU.

References

Barley, M. W.; Riddle, P. J.; Linares López, C.; Dobson, S.; and Pohl, I. 2018. GBFHS: A generalized breadth-first heuristic search algorithm. In *SoCS*, 28–36.

Burns, E. A.; Hatem, M.; Leighton, M. J.; and Ruml, W. 2012. Implementing fast heuristic search code. In *SoCS*.

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

de Champeaux, D., and Sint, L. 1977. An improved bidirectional heuristic search algorithm. *J. ACM* 24(2):177–191.

de Champeaux, D. 1983. Bidirectional heuristic search again. *J. ACM* 30(1):22–32.

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

Eckerle, J.; Chen, J.; Sturtevant, N. R.; Zilles, S.; and Holte, R. C. 2017. Sufficient conditions for node expansion in bidirectional heuristic search. In *ICAPS*.

Felner, A.; Korf, R. E.; and Hanan, S. 2004. Additive pattern database heuristics. *J. Artif. Intell. Res.* 22:279–318.

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic potential search - A new bounded suboptimal search. In *SoCS*, 36–44.

Helmert, M. 2010. Landmark heuristics for the pancake problem. In *SoCS*.

Holte, R. C.; Felner, A.; Sharon, G.; Sturtevant, N. R.; and Chen, J. 2017. MM: A bidirectional search algorithm that is guaranteed to meet in the middle. *Artif. Intell.* 252:232–266.

Holte, R. C. 2010. Common misconceptions concerning heuristic search. In Felner, A., and Sturtevant, N. R., eds., *SoCS*. AAAI Press.

Kaindl, H., and Kainz, G. 1997. Bidirectional heuristic search reconsidered. *J. Artificial Intelligence Resesearch (JAIR)* 7:283–317.

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

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

Russell, S. J., and Norvig, P. 2016. *Artificial Intelligence: A Modern Approach.* Malaysia; Pearson Education Limited,.

Shaham, E.; Felner, A.; Chen, J.; and Sturtevant, N. R. 2017. The minimal set of states that must be expanded in a front-to-end bidirectional search. In *SoCS*, 82–90.

Shaham, E.; Felner, A.; Sturtevant, N. R.; and Rosenschein, J. S. 2018. Minimizing node expansions in bidirectional search with consistent heuristics. In *SoCS*, 81–98.

Sharon, G.; Holte, R. C.; Felner, A.; and Sturtevant, N. R. 2016. Extended abstract: An improved priority function for bidirectional heuristic search. In *SoCS*, 139–140.

Shperberg, S.; Hayoun, A.; Felner, A.; Shimony, S. E.; and Sturtevant, N. R. 2019. Enriching non-parametric bidirectional search algorithms. In *AAAI*.

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