Front-to-End Bidirectional Heuristic Search with Consistent Heuristics: Enumerating and Evaluating Algorithms and Bounds

Lior Siag[1] , **Shahaf Shperberg**[1] , **Ariel Felner**[1] and **Nathan Sturtevant**[2,3]
[1]Ben-Gurion University of the Negev
[2]University of Alberta
[3]Alberta Machine Intelligence Institute (Amii)
siagl@post.bgu.ac.il, {shperbsh, felner}@bgu.ac.il, nathanst@ualberta.ca

Abstract

Recent research on bidirectional heuristic search (BiHS) is based on the *must-expand pairs* theory (MEP theory), which describes which pairs of nodes must be expanded during the search to guarantee the optimality of solutions. A separate line of research in BiHS has proposed algorithms that use lower bounds that are derived from consistent heuristics during search. This paper links these two directions, providing a comprehensive unifying view and showing that both existing and novel algorithms can be derived from the MEP theory. An extended set of bounds is formulated, encompassing both previously discovered bounds and new ones. Finally, the bounds are empirically evaluated by their contribution to the efficiency of the search.

1 Introduction

Research on bidirectional heuristic search (BiHS) dates back to the late sixties. Recently, a new theory was developed for BiHS [Eckerle *et al.*, 2017] that laid the foundation for a rich line of research in the area [Shaham *et al.*, 2017; Shaham *et al.*, 2018; Shperberg *et al.*, 2019a; Sturtevant *et al.*, 2020; Alcázar *et al.*, 2020; Alcázar, 2021; Shperberg *et al.*, 2021]. This theory of *must-expand pairs* (MEP) characterizes the set of forward- and backward- node pairs that must be expanded in order to prove solutions' optimality. Recent BiHS algorithms, such as NBS [Chen *et al.*, 2017] and DVCBS [Shperberg *et al.*, 2019b] utilize the MEP theory to reduce the search effort required to return optimal solutions.

This paper focuses on the case where the heuristic is known to be consistent. In this case, the conditions for which pairs of nodes must be expanded can be refined [Shaham *et al.*, 2018], which reduces the number of MEPs. Thus, assuming consistency, these tighter conditions can potentially be exploited to further reduce the number of nodes expanded by BiHS algorithms. Algorithms like NBS and DVCBS, however, cannot be efficiently adapted to these tighter conditions, whose enforcement requires significantly more computational effort.

BiHS algorithms for the consistent heuristics case have also been developed. These algorithms utilize the information derived from heuristic consistency to maintain lower bounds (denoted *search bounds*) on solutions' cost, which in turn are used for speeding up the search [Kaindl and Kainz, 1997; Sadhukhan, 2012; Alcázar *et al.*, 2020; Sewell and Jacobson, 2021]. Fundamental questions include whether other bounds can be developed and how well they will perform.

To answer these questions, the conceptual gap between these algorithms and the MEP theory must be explored. This paper provides a unifying view that closes this gap. The contributions of this paper are : **(1)** A family of search bounds that can be directly derived from the MEP conditions. To our knowledge, this family includes all existing search bounds as well as novel search bounds. **(2)** A global algorithmic framework designed to exploit any subset of bounds; with a focus on exploiting single bounds for guiding the search. This framework encompasses previous search-bounds-based algorithms. **(3)** An experimental study on the effect of each bound and the behaviors of variants of the framework. We also compare the number of nodes expanded by these variants to the theoretical lower bound from the MEP theory, showing that existing algorithms are very close to the theoretical limits.

2 Definitions, Notations, and Background

In BiHS, the aim is to find a least-cost path, of cost C^*, between $start$ and $goal$ in a given graph G. $dist(x, y)$ denotes the shortest distance between x and y, so $dist(start, goal) = C^*$. In some cases, the cost of the cheapest edge of the graph (denoted by ϵ) is available, otherwise, ϵ is assumed to be 0.

BiHS typically keeps two open lists, $Open_F$ for the forward search (F), and $Open_B$ for the backward search (B). Given a direction D (either F or B), we use f_D, g_D and h_D to indicate f-, g-, and h-values in direction D. We use $\overline{D}$ to denote the direction that is opposite to D, and define $f_{\overline{D}}$, $g_{\overline{D}}$ and $h_{\overline{D}}$ symmetrically. This paper considers the two *front-to-end* heuristic functions [Kaindl and Kainz, 1997] $h_F(s)$ and $h_B(s)$ which respectively estimate $dist(s, goal)$ and $dist(start, s)$ for all $s \in G$. h_F, is *forward admissible* iff $h_F(s) \leqslant dist(s, goal)$ for all s in G and is *forward consistent* iff $h_F(s) \leqslant dist(s, s') + h_F(s')$ for all s and s' in G. Backward *admissibility* and *consistency* are defined analogously.

In addition to the known g, h, and f functions, we define the following functions which are used throughout the paper:

(1) $d_D(n) = g_D(n) - h_{\overline{D}}(n)$, the *difference* between the actual cost in direction D of node n and its heuristic estimation; this indicates the *heuristic error* for node n.

(2) $b_D(n) = f_D(n) + d_D(n)$. $b_D(n)$ adds the heuristic error $d_D(n)$ to $f_D(n)$ to indicate that the opposite search using $h_{\overline{D}}(n)$ will underestimate by $d_D(n)$.
(3) $rf_D(n) = g_D(n) - h_D(n)$, the reverse f-value [Alcázar, 2021], a function that is similar to f, but which subtracts the heuristic instead of adding it.
(4) $rd_D(n) = g_D(n) + h_{\overline{D}}(n)$, the reverse d-value [Alcázar, 2021], which adds the opposite heuristic instead of subtracting it. The term *reverse* for rf and rd was coined by Alcázar *et al.* (2020). While the motivation behind rf and rd might seem unclear, they can be used as building blocks for some of the bounds on optimal solutions cost, as we show below.

Finally, we denote by $xMin_D$ the minimal x value in $Open_D$. For example, $gMin_D = \min_{n \in Open_D} g_D(n)$, the minimal g-value in direction D. Additionally, $\{x, y\}Min_D$ is the minimal $x + y$ value in $Open_D$, e.g., $\{rf, d\}Min_D = \min_{n \in Open_D}\{rf_D(n) + d_D(n)\}$. Likewise, $\{x, y, z\}Min_D$ is the minimal $x + y + z$ value in direction D.

2.1 Must-Expand Nodes in Bidirectional Search

Any unidirectional heuristic search (UniHS) algorithm (meeting reasonable theoretical assumptions), that is guaranteed to find an optimal solution on every problem with an admissible heuristic, *must expand* all nodes n with $f(n) < C^*$ to prove the optimality of solutions when given a consistent heuristic [Dechter and Pearl, 1985].

The generalization of this theory to BiHS [Eckerle *et al.*, 2017] showed that in BiHS the *must expand* attribute is defined on *pairs* of nodes (u, v) from the forward and backward frontiers (and not on a single node as in UniHS) as follows:

$$lb(u, v) = \max\{f_F(u), f_B(v), g_F(u) + g_B(v) + \epsilon\} \quad (1)$$

$lb(u, v)$ is a lower bound on the cost of any path that can connect $start$ and $goal$ via u and v. In BiHS, a pair of nodes (u, v) is called a *must-expand pair* (denoted MEP) if $lb(u, v) < C^*$. In a MEP at least one of u or v *must be expanded*. Otherwise, an algorithm that does not expand either u or v when $lb(u, v) < C^*$ might miss the optimal solution.

The set of MEPs can be reformulated as a bipartite graph, denoted as the *Must-Expand Graph* (GMX) [Chen *et al.*, 2017]. For each state $u \in G$, GMX includes a forward vertex u_F and a backward vertex u_B. For each pair of states $u, v \in G$, there is an edge in GMX between u_F and v_B iff (u, v) is a MEP. Since each edge of GMX is a MEP and at least one node from each MEP must be expanded to ensure the optimality of the solution, it follows that any optimal algorithm must expand a *vertex cover* of GMX. Thus, the minimum number of node expansions required to guarantee the optimality of a solution for problem instance by BiHS is the size of the *minimum vertex cover* (denoted by MVC) of GMX. When consistency is not assumed, the MVC of GMX can be computed in linear time with respect to the number of nodes in the GMX [Shaham *et al.*, 2017].

2.2 MEPs When Assuming a Consistent Heuristics

This paper focuses on the case where algorithms can assume that they are always given problem instances with a consistent heuristic (i.e., they do not need to return an optimal solution when given admissible heuristics that are not consistent). We denote this as the *consistency case*, under which more information can be exploited and the aim is to develop algorithms that utilize this. For the consistency case, Shaham *et al.* (2018) introduced tighter lower bounds for pairs denoted $lb_C(u, v)$ (C for consistency):

$$lb_C(u, v) = g_F(u) + g_B(v) + \max \begin{cases} h_F(u) - h_F(v) & \text{(2a)} \\ h_B(v) - h_B(u) & \text{(2b)} \\ \epsilon & \text{(2c)} \end{cases}$$

All terms in the max expression are lower bounds on $dist(u, v)$ and can thus be added to $g_F(u) + g_B(v)$. ϵ (term 2c) is a trivial lower bound, term 2a is derived from the definition of a forward-consistent heuristic: $h_F(u) \leqslant dist(u, v) + h_F(v) \implies h_F(u) - h_F(v) \leqslant dist(u, v)$, and term 2b is derived analogously from the definition of backward consistency.[1] $lb_C(u, v)$ induces a new MEP definition, denoted as MEP_C. Since $lb_C(u, v) \leqslant lb(u, v)$ for every pair of nodes u, v, the number of MEP_Cs is *smaller than or equal to* the number of MEPs for any given problem instance.

Finally, for undirected graphs, $lb_C(u, v)$ can be further tightened resulting in $lb_{CU}(u, v)$ (U for undirected):

$$lb_{CU}(u, v) = g_F(u) + g_B(v) + \max \begin{cases} h_F(u) - h_F(v) & \text{(3a)} \\ h_B(v) - h_B(u) & \text{(3b)} \\ \epsilon & \text{(3c)} \\ h_F(v) - h_F(u) & \text{(3d)} \\ h_B(u) - h_B(v) & \text{(3e)} \end{cases}$$

Both Equations 2 and 3 can be used for defining corresponding GMX graphs, however, in these cases the MVC can no longer be computed in linear time [Shaham *et al.*, 2018].

2.3 NBS and DVCBS

Near-Optimal Bidirectional Search (NBS) [Chen *et al.*, 2017] is a prominent algorithm based on the MEP theory. At every expansion cycle, NBS chooses a pair of nodes u, v with a minimal $lb(u, v)$ (denoted as LB) among all pairs in the open lists, and expands *both* nodes. This approach is based on the 2-factor approximation of minimal vertex covers [Papadimitriou and Steiglitz, 1998]. Thus, NBS is guaranteed to expand a vertex cover of GMX whose size is at most $2 \times |MVC|$. To find a pair with minimal lb value, NBS uses two-level priority queues in each direction, a *waiting queue* which stores all nodes n with $f_D(n) > LB$, sorted by f, and *ready queue* which stores all nodes n with $f_D(n) \leqslant LB$ sorted by g. These data structures enable NBS to only expand MEPs, while maintaining amortized insertion and deletion time of $\log(n)$, where n is the number of frontier nodes.

Dynamic Vertex Cover Bidirectional Search (DVCBS) [Shperberg *et al.*, 2019b] is a competitor to NBS. DVCBS constructs a partial, dynamic GMX (denoted DGMX) based on the nodes currently in OPEN, finds an MVC of DGMX, and expands all nodes in this MVC. DVCBS is unbounded in the worst case, but empirically outperforms NBS on average. DVCBS computes the MVC of DGMX by bucketing nodes

[1] Equation 1 for $lb(u, v)$ includes terms for $f_F(u)$ and $f_B(v)$. These terms are redundant in $lb_C(u, v)$, as $g_F(u)+g_B(v)+h_F(u)-h_F(v) = f_F(u)+d_B(v) > f_F(u)$, and for $f_B(v)$ using term 2b.

with the same f_D- and g_D-values and solving a weighted MVC problem in time that is *linear* in the number of buckets.

An issue arises when attempting to adapt NBS and DVCBS to the consistency case. Although NBS can be defined for lb_C (Equation 2) and lb_{CU} (Equation 3), its two-level priority queue is no longer suitable for efficiently identifying a pair with the minimum lb_C (or lb_{CU}) value. Similarly, DVCBS can no longer achieve a linear-time MVC of DGMX. In an effort to develop an efficient algorithm for the consistency case, Alcázar (2021) introduced a method called bucket-to-bucket (BTB). This scheme involves grouping nodes into buckets that share the same $g_D, h_D, h_{\overline{D}}$ values. Using these buckets, both NBS and DVCBS can be applied using lb_C (or lb_{CU}). However, each expansion cycle of a bucket requires a computational overhead that is quadratic in the number of buckets. While the number of buckets may be small in certain domains, in general, there could be an arbitrary number of buckets. As a result, the quadratic runtime for every expansion cycle can become impractical (e.g., in road maps or grids).

2.4 Search Bounds, DIBBS/BAE*, and DBBS

We use the term *search bounds* [Alcázar *et al.*, 2020] to denote all lower bounds on the solution cost that can be computed during the search. We classify the search bounds into three categories. **(1)** *Global bounds* provide a lower bound on the cost of every solution from $start$ to $goal$. These bounds can be used as termination conditions; when the incumbent solution has a cost that equals the bound then it is optimal. **(2)** *Individual-node bounds* bound the cost of every solution from $start$ to $goal$ that passes through a given node n (e.g. $lb(n)$). **(3)** *Pair bounds* bound the cost of paths from $start$ to $goal$ that passes through a given pair of nodes u, v, where $u \in Open_F$ and $v \in Open_B$ (e.g., $lb(u, v)$).

Several search bounds have been used in the heuristic search literature. In UniHS the minimal f-value in the open list, $fMin$, is a global bound on the cost of any solution. This f*-bound* is commonly used in the termination conditions of many algorithms (both unidirectional and bidirectional). Similarly, in BiHS, $fMin_F$ and $fMin_B$ are global bounds. Another bound in BiHS is the g*-bound*$= gMin_F + gMin_B + \epsilon$.

For the consistency case, other search bounds have been proposed. Kaindl and Kainz (1997) proposed adding heuristic errors (defined as $d_D(n)$ above), of direction D, to f-values of the opposite direction $\overline{D}$. In particular, they defined the following individual-node bounds $KKAdd_D(n) = f_D(n) + dMin_{\overline{D}}$ and symmetrically, $KKMax_D(n) = d_D(n) + fMin_{\overline{D}}$. Note that these node bounds can be transformed to be global bounds by taking the minimal f- and d-values from both directions as follows: $KKAdd = fMin_D + dMin_{\overline{D}}$ and $KKMax = dMin_D + fMin_{\overline{D}}$. A lower bound b' is said to *dominate* another lower bound b'' iff $b' \geqslant b''$. Since $fMin_D + dMin_{\overline{D}} \geqslant fMin_F$ then $KKAdd$ dominates $fMin_F$. Similarly, $KKMax$ dominates $fMin_B$. Another global bound is the foundation of two identical algorithms, BAE* [Sadhukhan, 2012] and DIBBS [Sewell and Jacobson, 2021], which expand nodes with minimal b-value. They introduce a new global bound, b-bound$= {}^{(bMin_F + bMin_B)}/_2$, and terminate when a solution is found with cost $= b$-bound.

Alcázar (2021) combined the KK bounds, the g-bound, and the b-bound together to improve their individual performance. In this context, a node n is defined as *expandable* iff $n \in \operatorname{argmin}_{n' \in Open_D}(\max\{f_D(n') + dMin_{\overline{D}}, d_D(n') + fMin_{\overline{D}}, {}^{(b_D(n') + bMin_{\overline{D}})}/_2, g_D(u) + gMin_{\overline{D}} + \epsilon\}$. For an undirected graph, two more bounds were introduced: $rfMin_F + rdMin_B$ (forward rc (reverse consistent)) and $rdMin_F + rfMin_B$ (backward rc). The minimal values in the open lists, which are used for defining the global search bounds, can be computed with respect to the set of expandable nodes. This creates a *fixpoint computation* in which the minimal values in the open lists are updated based on the set of expandable nodes, and the set of expandable nodes is updated based on the updated minimal values in the open lists until convergence is reached. The DBBS algorithm [Alcázar, 2021] performs this fixpoint computation to find tighter bounds, and thus expand fewer nodes during the search. This process, however, is computationally expensive.

3 Deriving Bounds from the MEP Theory

In this section, we introduce a unifying view for the consistency case and draw a connection between lb_C (Equation 2) and all existing search bounds (presented in Section 2.4). In addition, we show that additional search bounds can be derived from lb_C and even more bounds can be derived when also considering undirected graphs (lb_{CU}, Equation 3).

A first step towards a unifying view was taken by Alcázar (2021) who observed that taking each element in the max term of Equation 2 (2a, 2b, and 2c) individually, can derive the $KKAdd$, $KKMax$, and g-bound, correspondingly. For example, for term 2a, $g_F(u) + g_B(v) + h_F(u) - h_F(v) = f_F(u) + d_B(v) \geqslant fMin_F + dMin_B = KKAdd$. However, the method of taking individual elements of the max term is limited and cannot be used for producing other bounds, e.g., the b-bound. We now provide a general way to produce more bounds from the max term. In fact, any max function over a set of elements S can be bounded from below as follows:

$$\max S \geqslant \frac{\sum_{s' \in S'} s'}{|S'|} \qquad \forall S' \in \mathcal{P}(S) \backslash \varnothing \tag{4}$$

where $\mathcal{P}(S)$ is the power set of S (i.e., the average of the elements of any subset of S is always a lower bound of the maximal element in S). Using this rule, the max expression in Equation 2 can be lower-bounded by different subsets S' of the terms inside (i.e., 2a, 2b, and 2c). For example, when taking $S' = \{2a, 2b\}$, we get the b-bound (B_4 below):

$$\begin{aligned} lb_C(u, v) &\geqslant g_F(u) + g_B(v) + \frac{[\text{term } 2a] + [\text{term } 2b]}{2} \\ &= \tfrac{2g_F(u) + 2g_B(v) + h_F(u) - h_F(v) + h_B(v) - h_B(u)}{2} \\ &= {}^1/_2 \cdot (f_F(u) + d_F(u) + f_B(v) + d_B(v)) \\ &= {}^1/_2 \cdot (b_F(u) + b_B(v)) \geqslant {}^1/_2 \cdot (bMin_F + bMin_B) \\ &= b\text{-bound } (B_4) \end{aligned}$$

By considering all subsets S' of terms 2a, 2b, and 2c, and for a node n by replacing the value $x_D(n)$ with the minimal x-value in $Open_D$, $xMin_D$, as was demonstrated above (for $2a$ and for $2a, 2b$), we get the following global bounds:

B_1: $fMin_F + dMin_B$ ($KKAdd$), when $S' = \{2a\}$

B_2: $dMin_F + fMin_B$ ($KKMax$), when $S' = \{2b\}$

B_3: $gMin_F + gMin_B + \epsilon$ (g-bound), when $S' = \{2c\}$

B_4: $\frac{bMin_F + bMinB}{2}$ (b-bound), when $S' = \{2a, 2b\}$

B_5: $\frac{\{f,g\}Min_F + \{d,g\}Min_B + \epsilon}{2}$, when $S' = \{2a, 2c\}$

B_6: $\frac{\{d,g\}Min_F + \{f,g\}Min_B + \epsilon}{2}$, when $S' = \{2b, 2c\}$

B_7: $\frac{\{b,g\}Min_F + \{b,g\}Min_B + \epsilon}{3}$, when $S' = \{2a, 2b, 2c\}$

The derivation of each of the bounds $\{B_1, \ldots, B_7\}$ can be done similarly to the two examples presented above.

We note that substantial work was done to individually prove each of the previously proposed bounds (e.g., [Kaindl and Kainz, 1997; Sewell and Jacobson, 2021]), whereas the derivation of these bounds from the MEP theory is straightforward. Finally, note that individual-node bounds that correspond to each of the above global bounds $\{B_1, \ldots, B_7\}$ can be derived from the MEP theory as well. This is done by replacing the value of $x_D(n)$ with the minimal x value in $Open_D$ ($xMin_D$) in one of the two directions instead of the two directions, as was done in the derivations above. For example, the last step in the $KKAdd$ derivation above (B_1) becomes $f_F(u) + d_B(v) \geqslant f_F(u) + dMin_B$.

3.1 Bounds for Undirected Graphs

For undirected graphs, we derive lower bounds from lb_{CU} instead of lb_C. Since terms 2a, 2b, and 2c are identical to 3a, 3b, and 3c, the above seven bounds $\{B_1, \ldots, B_7\}$ are also valid for for lb_{CU}. Nevertheless, additional bounds for the max expression can be derived from terms 3d and 3e. At first glance, it might seem that there are 31 possible search bounds for lb_{CU} (corresponding to the 31 possible subsets of terms 3a to 3e excluding $\varnothing$). However, note that terms 3a and 3d cancel each other ($3a + 3d = h_F(u) - h_F(v) + h_F(v) - h_F(u) = 0$). Thus, all subsets S' that contain both term 3a and term 3d are dominated either by $S' \backslash \{3a\}$ or by $S' \backslash \{3d\}$ (which are also valid subsets of S), and can be ignored. The same argument holds for terms 3b and 3e. Thus, we are left with only 17 bounds that are not dominated by other bounds. The resulting (additional) bounds and their respective subsets of terms are as follows:

B_8: $rfMin_F + rdMin_B$ ($forward\ rc$), when $S' = \{3d\}$

B_9: $rdMin_F + rfMin_B$ ($backward\ rc$), when $S' = \{3e\}$

B_{10}: $\frac{\{f,rd\}Min_F + \{rf,d\}Min_B}{2}$, when $S' = \{3a, 3e\}$

B_{11}: $\frac{\{rf,d\}Min_F + \{f,rd\}Min_B}{2}$, when $S' = \{3b, 3d\}$

B_{12}: $\frac{\{rf,g\}Min_F + \{rd,g\}Min_B + \epsilon}{2}$, when $S' = \{3c, 3d\}$

B_{13}: $\frac{\{rd,g\}Min_F + \{rf,g\}Min_B + \epsilon}{2}$, when $S' = \{3c, 3e\}$

B_{14}: $\frac{\{rf,rd\}Min_F + \{rf,rd\}Min_B}{2}$, when $S' = \{3d, 3e\}$

B_{15}: $\frac{\{f,rd,g\}Min_F + \{rf,d,g\}Min_B + \epsilon}{3}$, when $S' = \{3a, 3c, 3e\}$

B_{16}: $\frac{\{rf,d,g\}Min_F + \{f,rd,g\}Min_B + \epsilon}{3}$, when $S' = \{3b, 3c, 3d\}$

B_{17}: $\frac{\{rf,rd,g\}Min_F + \{rf,rd,g\}Min_B + \epsilon}{3}$, when $S' = \{3c, 3d, 3e\}$

The above set of bounds ($\{B_1, \ldots, B_7\}$ and $\{B_1, \ldots, B_{17}\}$ for Equation 2 and Equation 3 (respectively) are all the possible bounds that can be produced when approximating the max term in the equations using Equation 4. Other lower bounds would require to bound the max terms in different ways (possibly using a convex combination) or using other methods altogether. Another possibility for generating new bounds is to introduce additional assumptions. For instance, as shown by Alcázar *et al.* (2020), if the greatest common denominator among the cost of all non-zero-cost edges, denoted as ι, is known, it can be used to tighten the bounds further. For each bound B_i from the above set of bounds, we can use $\iota\lceil \frac{B_i}{\iota} \rceil$ as a possibly tighter version of B_i. So, in unit-edge-cost graphs, all of the above bounds can be rounded up.

4 Using the Bounds within Algorithms

We now turn to the practical aspects of the proposed bounds. We describe a general BiHS algorithmic framework for the consistency case, presented in Algorithm 1, and show how to utilize information from the bounds in this framework. There are three decisions that define BiHS algorithms: (1) deciding when to terminate the search, (2) choosing which node to expand from a given direction, and (3) choosing the direction from which the next node will be expanded. Each of these can utilize information from the bounds.

4.1 Termination Condition

Each global bound B_i can be used to prove that the incumbent solution is optimal, i.e., when a solution is found with $cost = B_i$ the algorithm can terminate. Naturally, when there are several lower bounds, their maximum is the tightest lower bound among them and can be used as a termination condition. The advantage of using several bounds is that if we have the optimal solution U (with cost C*) in hand, we can halt faster: once $B_i = C$* for any of the available global bound B_i. The tradeoff is the overhead of maintaining and consulting several bounds. In addition, the bounds can be computed either with the (computationally expensive) fixpoint computation, as in DBBS, or without it.

4.2 Choosing Which Node to Expand

The decision of which node in $Open_D$ to expand next (i.e., what priority function to use) is at the heart of any search algorithm. Since each global bound B_i is essentially a termination condition, expanding nodes that will cause B_i to increase as early as possible would likely result in earlier termination. For example, A* [Hart *et al.*, 1968] terminates when a solution is found whose cost equals the minimal f-value in $Open_F$ ($fMin_F$). Consequently, A* expands a node n with $f(n) = fMin_F$, as this expansion policy is *targeted* at increasing the f-bound. Similarly, BAE* and DIBBS used the b-bound as a termination condition and thus expand nodes n with minimal $b_D(n)$ value. In a similar manner, expansion policies can be defined to target each of the above bounds. For example, the expansion policy that focuses on increasing $KKAdd$ (bound B_1) expands a node n with minimal $f_F(n)$-value or the node m with minimal $d_B(m)$, depending on the chosen direction D. In general, each of our bounds has a

Algorithm 1: BiHS General Algorithmic Framework

1 $U \leftarrow \infty, LB \leftarrow \mathit{ComputeLowerBound}()$
2 **while** $\mathit{Open}_F \neq \varnothing \wedge \mathit{Open}_B \neq \varnothing \wedge U > LB$ **do**
3 $\quad D \leftarrow \mathit{ChooseDirection}()$
4 $\quad n \leftarrow \mathit{ChooseNode}(D)$
5 $\quad \mathit{Expand}(n, D)$ `// update` U
6 $\quad LB \leftarrow \mathit{ComputeLowerBound}()$
7 **return** U

term to minimize from the forward side and a term to minimize from the backward side, and we choose to expand a node whose bound equals the relevant Min term according to the chosen direction D.

When several global bounds are used, we need to choose which bound to target. Many policies are possible; we describe two of them that we used in our experiments:

Targeting the max. This policy targets the bound which has the maximal value among all bounds, B_{max}. The motivation here is that if we manage to raise this bound by expanding nodes, then we have a higher chance that the incumbent solution U will have $cost(U) = B_{max}$, and the algorithm can terminate. Thus, we choose to expand a node n in a given direction D whose individual bound $B_{max}(n) = B_{max}$.

Targeting the bound with the smallest span. Given a global bound B, we define $span(B)$ to be the number of nodes n with an individual-node bound $B(n) = B$. Let B_{ss} be the global bound with the *smallest span* in the set of bounds. This policy chooses to expand a node n whose individual bound $B_{ss}(n) = B_{ss}$. The intuition for this policy is to focus on the bound that would require the smallest effort to increase.[2]

4.3 Choosing Which Side to Expand

Finally, we deal with the decision of whether to expand a node from $Open_F$ or $Open_B$ at each step. To this end, we explore three policies: *alternating*, *cardinality criterion* [Pohl, 1971], and a new policy called *fastest bound increase (FBI)*. The alternating policy switches between forward and backward sides, while the cardinality criterion selects the side with the smallest open list. FBI aims to achieve the fastest increase in a specific bound B_i by comparing the spans of B_i in $Open_F$ and $Open_B$ and choosing the direction with the smaller span. For policies targeting the maximal bound or minimal span, FBI calculates B_{max} or B_{ss} for both directions and selects the side with the smallest span.

5 Empirical Evaluation

The main purpose of the empirical evaluation is to assess the different bounds. To this end, we implemented a **targeted bound** algorithm for each of the 17 bounds, denoted as TB_i for all $1 \leqslant i \leqslant 17$. In TB_i we use all 17 bounds for termination (Line 3 in Algorithm 1), but only the targeted bound B_i for choosing which node to expand (Line 5 in Algorithm 1).

[2]This is a greedy computation based on the current state of the open lists, as new nodes with the same B_{ss}-values can be generated.

A secondary objective is to compare the performance of the new algorithms with notable existing algorithms and to determine when each algorithm should be used. For this, we evaluated **BAE***, which uses only the b-bound for both node expansion and as the termination condition (related to TB_4 that expands nodes with minimal b-value, but terminates using all the bounds). We evaluated both Pohl's cardinality criterion (denoted (p)) and alternating (denoted (a)) as side-selection policies (Line 4 in Algorithm 1) for BAE* and the TB_i algorithms. We also experimented with **A*** and **reverse A*** (denoted as rA*), which performs the search from $goal$ to $start$ as representatives of UniHS algorithms. In addition, for determining the effect of the consistency assumption on the search we experimented with **NBS** and **DVCBS**, which assume heuristic admissibility, but not consistency. Finally, to study the effect of the fixpoint computation on the new bounds, we ran DBBS, which uses the original four global bounds, as well as DBBS^{all}, which uses all 17 global bounds. For the DBBS variants, we evaluate the node-expansion policy that targets the b-bound (as in [Alcázar *et al.*, 2020]), and the new node-expansion policies, max and smallest span. For side-selection policies, we use (a), (p), and (FBI).

Finally, we calculated the theoretical lower bound on the number of expansions required to guarantee the optimality of solutions. To this end, we compute and report the MVC of GMX using lb (Equation. 1), in which only heuristic admissibility is assumed, as well as the MVC of GMX_{CU} (defined according to lb_{CU}, Equation. 3), in which the consistency is also assumed. This is the first time that GMX_{CU} has been computed in the BiHS literature. To compute the MVC, we constructed GMX_{CU} post facto (by collecting all nodes with $f_D(n) < C^*$ to buckets), reduced the problem to a max-flow problem (König's theorem [Konig, 1931]), and ran the Ford–Fulkerson algorithm [Ford and Fulkerson, 1956] to find the maximal flow, which is equal to the MVC of GMX_{CU}. The difference between the MVC and the actual number of nodes expanded by the evaluated algorithms indicates whether new BiHS algorithms that assume consistency should be developed (if there is a significant gap between them) or the focus should be turned elsewhere.

5.1 Experimental Settings

Domains. We experimented on five domains: **(1)** 50 14-pancake puzzle instances with the GAP heuristic [Helmert, 2010]. To get a range of heuristic strengths, we also used the GAP-n heuristics (for $1 \leqslant n \leqslant 6$) where the n smallest pancakes are left out of the heuristic computation. **(2)** The standard 100 instances of the **15 puzzle** problem (STP) [Korf, 1985] using the Manhattan distance (MD) heuristic. **(3) Grid**-based pathfinding using octile distance as a heuristic and 1.5 cost for diagonal edges: 156 maps from Dragon Age Origins (DAO) [Sturtevant, 2012], each with different start and goal points (a total of 3149 instances); **(4)** 50 instances of the 12-disk **4-peg Towers of Hanoi** (TOH) problem with (10+2), (8+4) and (6+6) additive PDBs [Felner *et al.*, 2004]; and, **(5)** 100 random **road map** instances [Demetrescu *et al.*, 2009] on the map of Colorado, using the Euclidean distance divided by the maximum speed as a heuristic.

		ToH - 12						DAO		Road Maps		STP	
		PDB (10+2)		PDB (8+4)		PDB (6+6)		Octile		Dist/Speed$_{max}$		MD	
	Algorithm	$\leqslant C^*$	$< C^*$	$\leqslant C^*$	$< C^*$	$\leqslant C^*$	$< C^*$	$\leqslant C^*$	$< C^*$	$\leqslant C^*$	$< C^*$	$\leqslant C^*$	$< C^*$
Admissible	A*	276K	276K	1,926K	1,925K	3,268K	3,239K	5,406	5,322	127K	127K	15,550K	14,700K
	rA*	123K	123K	622K	620K	1,064K	1,033K	**5,342**	5,267	126K	126K	11,144K	10,956K
	NBS	230K	230K	647K	645K	682K	662K	6,873	6,556	96K	96K	12,556K	11,739K
	DVCBS	232K	232K	619K	609K	663K	635K	5,545	5,151	126K	126K	10,930K	10,720K
	MVC(GMX)		123K		557K		624K		4,290		78K		9,267K
Consistent	MVC(GMX$_{CU}$)		41K		174K		363K		N/A		N/A		1,308K
	BAE*(a)	**47K**	**46K**	**187K**	**186K**	**383K**	382K	6,718	6,668	**67K**	**67K**	2,707K	2,700K
	TB$_1$(a)	54K	53K	204K	204K	424K	423K	7,081	6,940	70K	70K	13,404K	12,953K
	TB$_2$(a)	69K	69K	240K	240K	435K	435K	7,150	7,018	72K	72K	10,815K	10,677K
	TB$_3$(a)	648K	622K	662K	660K	683K	664K	10,275	8,654	119K	119K	N/A	N/A
	TB$_4$(a)	**47K**	**46K**	**187K**	**186K**	**383K**	382K	6,706	6,483	**67K**	**67K**	2,707K	2,700K
	TB$_5$(a)	263K	262K	386K	382K	515K	509K	9,213	7,955	96K	96K	N/A	N/A
	TB$_6$(a)	282K	277K	412K	406K	513K	512K	9,221	8,021	96K	96K	N/A	N/A
	TB$_7$(a)	170K	165K	315K	308K	458K	455K	8,562	7,563	86K	86K	13,448K	13,275K
	DBBS$_b$(a)	48K	**46K**	189K	**186K**	**383K**	**379K**	6,105	5,829	N/A	N/A	2,262K	1,701K
	DBBS$_{max}$(FBI)	49K	48K	200K	197K	407K	400K	5,649	5,369	N/A	N/A	2,521K	1,761K
	DBBS$_{ss}$(FBI)	48K	47K	194K	192K	395K	393K	**5,374**	**4,935**	N/A	N/A	**2,027K**	**1,677K**
	DBBS$_b^{all}$(a)	48K	**46K**	189K	**186K**	**383K**	**379K**	6,104	5,827	N/A	N/A	2,262K	1,701K
	DBBS$_{max}^{all}$(FBI)	59K	55K	236K	233K	549K	527K	5,648	5,367	N/A	N/A	3,057K	2,208K
	DBBS$_{ss}^{all}$(FBI)	50K	48K	195K	194K	398K	397K	5,595	5,181	N/A	N/A	2,054K	1,705K

Table 1: Average number of nodes expanded for ToH, DAO, Road Maps, and STP

		GAP		GAP-2		GAP-4	
	Algorithm	$< C^*$	$\leqslant C^*$	$< C^*$	$\leqslant C^*$	$< C^*$	$\leqslant C^*$
Admissible	A*	**72**	46	351K	348K	47,289K	N/A
	rA*	77	52	350K	349K	48,323K	N/A
	NBS	148	66	123K	122K	1,433K	1,433K
	DVCBS	209	47	85K	84K	977K	899K
	MVC(GMX)		39		82K		N/A
Consistent	MVC(GMXCU)		37		6,711		N/A
	BAE*(a)	90	66	15K	11K	469K	465K
	TB$_1$(a)	136	86	78K	78K	5,581K	5,492K
	TB$_2$(a)	149	100	81K	79K	6,178K	6,036K
	TB$_3$(a)	484K	120K	1,508K	1,508K	1,623K	1,623K
	TB$_4$(a)	90	66	15K	11K	469K	465K
	TB$_5$(a)	27K	16K	150K	133K	674K	634K
	TB$_6$(a)	26K	16K	153K	142K	741K	658K
	TB$_7$(a)	2,724	2,073	35K	33K	301K	282K
	DBBS$_b$(a)	114	57	10K	8,801	277K	272K
	DBBS$_{max}$(FBI)	329	46	18K	10K	447K	378K
	DBBS$_{ss}$(FBI)	104	**43**	8,545	7,678	232K	230K
	DBBS$_b^{all}$(a)	114	57	10K	8,801	274K	269K
	DBBS$_{max}^{all}$(FBI)	307	46	21K	11K	394K	310K
	DBBS$_{ss}^{all}$(FBI)	104	**43**	**8,410**	**7,562**	**223K**	**219K**

Table 2: Results on the 14-Pancake Problem

Metrics. For each algorithm, we report the average number of node expansions required to **terminate** (denoted as $\leqslant C^*$). In addition, we report the average number of necessary expansions required to **prove optimality** (denoted as $< C^*$), i.e., the number of nodes expanded until LB has reached C^*.

Hardware. We ran the experiment on an 82-machine cluster. Each combination of algorithm and problem instance had a memory limit of 128GB and a time limit of 24 hours. When algorithms failed to solve some problems due to time or memory limits, the results are reported as N/A.

5.2 Results

Due to space limitations, only the results for bounds 1-7 (without the undirected graphs assumption) are included, as bounds 8-17 were shown to be weak. Moreover, results for the cardinality (p) side-selection policy are not reported, as there were no significant differences compared to the alternating policy. The results for ToH, DAO, road maps, and STP appear in Table 1 and for the pancake problem in Table 2.

Comparison to the MVC of the GMXs

In ToH, STP, and pancake the MVC of GMX is significantly larger than the MVC of GMX$_{CU}$ (by a factor of 3 for ToH, 7 for STP, and 8-15 for pancake, except for GAP, which is very accurate). This shows that the consistency assumption significantly reduces the number of necessary expansions in these domains. In road maps and DAO the MVC of GMX$_{CU}$ was too costly to compute due to large solution costs and many unique buckets. Nonetheless, the performance of the algorithms that only assume admissibility compared to algorithms that also make the consistency assumption hint that the MVC of GMX and GMX$_{CU}$ have a similar size in DAO, while the MVC of GMX$_{CU}$ is smaller in road maps by a small margin.

Comparison of Node Expansions

In **ToH** and **road maps**, BAE* had the best performance among all algorithms, similar to DBBS$_b^{all}$ (except for road maps, in which the DBBS variants could not complete execution due to the 24h runtime limit). Moreover, the difference between the number of nodes expanded by BAE* and the MVC of GMX$_{CU}$ is very small ($5-10\%$). Therefore, BAE* is very close to optimal, and no other algorithm (UniHS or BiHS) will be able to significantly improve over it. Finally, we see that the performance of BAE* is identical to TB$_4$, which means that the b-bound was always the tightest and never benefited from the other termination criteria.

In **STP**, BAE* and TB$_4$ were the best among the bounds that do not make use of fixpoint computation, improving over A*, rA*, NBS and DVCBS by a factor > 4. Here too, the additional stopping criteria of TB$_4$ did not improve the performance over BAE*. The DBBS variants, which perform the fixpoint computation, improved over BAE* by approximately

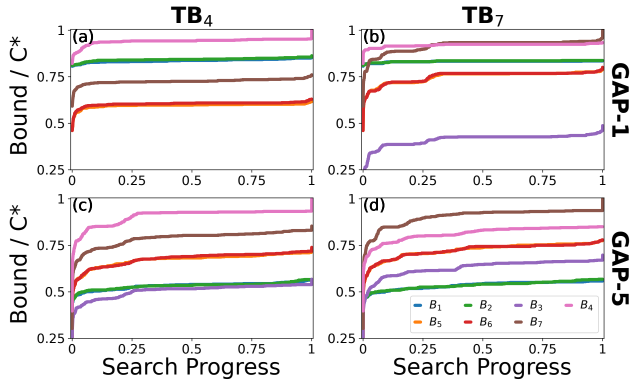

Figure 1: Value of different bounds as a function of search progress

25%, where the best-performing variant was DBBS$_{ss}$(FBI). In fact, DBBS$_{ss}$(FBI) expands only 28% more nodes than the MVC of GMX$_{CU}$ before raising LB to C^*. Thus, there is only a small margin left for improvement in STP as well.

In contrast to ToH, STP, and road maps, BAE* does not perform well on **DAO**. In this domain, rA* required the least expansions to return a solution, and DBBS$_{ss}$(FBI) required only a few additional expansions ($\leqslant C^*$), and fewer necessary expansions ($< C^*$). Notably, the additional termination conditions of TB$_4$ show a (small) improvement over BAE*, which means that the b-bound is not always dominating in DAO, even when the node-expansion policy only targets b.

Finally, in **pancake** (Table 2), we see a slightly different trend. A* is the best-performing algorithm for GAP, which is a very accurate heuristic, though DBBS$_{ss}$(FBI) required fewer necessary expansions. Among the algorithms that do not perform the fixpoint computation, BAE* had the best performance on GAP-1 and GAP-2, while TB$_7$ (which also uses the g-value of nodes, as well as ϵ) had a better performance (by up to a factor of 2) on GAP-3 to GAP-5, which are weaker heuristics. When also considering algorithms that perform the fixpoint computation, the new DBBS$_{ss}^{all}$(FBI) has the best performance for GAP-1 to GAP-5, improving over the best non-fixpoint algorithm by up to 40%, and getting close to the theoretical bound (MVC), with a difference of at most 15%.

The Effect of Bounds throughout the Search

We now turn to analyze the effect of the TB$_i$ variants for different values of i on the value of all used bound throughout the search. Figure 1 shows the average value of each of the 7 bounds (as a fraction of C^*) during the search on the pancake domain using two targeted bounds, TB$_4$ (b-bound, left) and TB$_7$ (right), and two heuristics, GAP-1 (top) and GAP-5 (bottom). The bound on top in each plot is the dominating bound, and the search terminates when this dominating bound equals C^*. Figures 1c and 1d show that on GAP-5, the bound that is targeted is the one with the highest value throughout the search, indicating that the targeted-bound policy achieves its purpose of rapidly increasing the bound of interest. However, Figure 1b shows that when one of the non-targeted bounds is significantly better than the targeted bound, the non-targeted bound might still have the highest value during the search.

	STP		DAO	
Algorithm	**Time (s)**	n/s	**Time (ms)**	n/ms
A*	55.42	281K	2.28	2371.12
BAE*(a)	10.07	269K	4.08	1646.61
TB$_4$(a)	59.99	45K	87.69	76.48
DBBS$_b$(a)	34.85	65K	2,390.74	2.55

Table 3: Runtime Results of Representative Algorithms

Runtime Analysis

Table 3 shows a focused report of the runtime for representative algorithms and problem instances. We used DAO and STP as polynomial and exponential representative domains, respectively. We aim to answer the following questions: (1) What is the overhead of using all bounds for termination compared to using a single bound? (2) What is the overhead of the fixpoint computation? and (3) How does the overhead of the BiHS algorithms compare to A*? The results show that BAE* is between 6 and 20 times faster than TB$_4$, due to the overhead required to maintain all other bounds (a sorted vector of pointers for each bound). This suggests that TB$_4$ should never be used, as its greater runtime overhead does not justify the small reduction in node expansions. As for the second question, the runtime of DBBS (i.e., the fixpoint computation) in DAO (polynomial domain) is several orders of magnitude smaller than other algorithms, while in STP (exponential domain) DBBS is only four times slower in terms of node expansions per second. Nonetheless, even in STP we see that BAE* is faster than DBBS in terms of total runtime, despite expanding more nodes. Finally, we see that A* is only slightly faster than BAE* in terms of expansions per second, thus the total runtime of the two algorithms mostly depends on their respective number of expansions.

As a general guideline, based on the empirical evaluation, BAE* should be the default algorithm to use for the consistency case, as it strikes a good balance between node expansions and runtime across different domains and heuristics. For weaker heuristics (e.g., as seen in GAP-4), we suggest using a targeted-bound algorithm that uses only B_7. While in some domains the DBBS variants expand the least number of nodes, the overhead of the fixpoint computation is not justified in terms of total runtime.

6 Summary and Conclusions

This paper drew a connection between the MEP theory and the existing search bounds, showing that all bounds can be directly derived from the theory. Furthermore, we introduced a set of 17 search bounds, both existing and novel. Algorithms that target each of these bounds were developed and evaluated, showing that the b-bound is the most informative bound for most domains, but another bound (B_7, which uses both b and g) is better for weak heuristics. In addition, the DBBS variants often expand the smallest number of nodes, but they incur an expensive computation overhead that always resulted in larger runtime. Finally, we compared the actual expansions by algorithms to the optimal number of necessary expansions, showing that there is not much room for further improvement. So, the focus on BiHS research can now move away from devising more algorithms for the consistency case.

Acknowledgements

This research was supported by the US-Israel Binational Science Foundation (BSF) grant no. 2021643. This work was also funded by the Canada CIFAR AI Chairs Program. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[Alcázar *et al.*, 2020] Vidal Alcázar, Patricia J. Riddle, and Mike Barley. A unifying view on individual bounds and heuristic inaccuracies in bidirectional search. In *AAAI*, pages 2327–2334. AAAI Press, 2020.

[Alcázar, 2021] Vidal Alcázar. The consistent case in bidirectional search and a bucket-to-bucket algorithm as a middle ground between front-to-end and front-to-front. In *ICAPS*, pages 7–15. AAAI Press, 2021.

[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 *IJCAI*, pages 489–495. ijcai.org, 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.

[Demetrescu *et al.*, 2009] Camil Demetrescu, Andrew V Goldberg, and David S Johnson. *The shortest path problem: Ninth DIMACS implementation challenge*, volume 74. American Mathematical Soc., 2009.

[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*, pages 79–87, 2017.

[Felner *et al.*, 2004] Ariel Felner, Richard E Korf, Ram Meshulam, and Robert C Holte. Compressing pattern databases. In *National Conference on Artificial Intelligence (AAAI-04)*, pages 638–643, 2004.

[Ford and Fulkerson, 1956] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[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 Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968.

[Helmert, 2010] Malte Helmert. Landmark heuristics for the pancake problem. In *SoCS*, pages 109–110. AAAI Press, 2010.

[Kaindl and Kainz, 1997] Hermann Kaindl and Gerhard Kainz. Bidirectional heuristic search reconsidered. *J. Artif. Intell. Res.*, 7:283–317, 1997.

[Konig, 1931] Dénes Konig. Graphok es matrixok (hungarian)[graphs and matrices]. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

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

[Papadimitriou and Steiglitz, 1998] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[Pohl, 1971] Ira Pohl. Bi-directional search. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6, pages 127–140. Edinburgh University Press, 1971.

[Sadhukhan, 2012] Samir K. Sadhukhan. A new approach to bidirectional heuristic search using error functions. In *CSI*, 2012.

[Sewell and Jacobson, 2021] Edward C. Sewell and Sheldon H. Jacobson. Dynamically improved bounds bidirectional search. *Artif. Intell.*, 291:103405, 2021.

[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. AAAI Press, 2018.

[Shperberg *et al.*, 2019a] Shahaf S. Shperberg, Ariel Felner, Solomon Eyal Shimony, Nathan R. Sturtevant, and Avi Hayoun. Improving bidirectional heuristic search by bounds propagation. In *SoCS*, pages 106–114, 2019.

[Shperberg *et al.*, 2019b] Shahaf S. Shperberg, Ariel Felner, Nathan R. Sturtevant, Solomon Eyal Shimony, and Avi Hayoun. Enriching non-parametric bidirectional search algorithms. In *AAAI*, pages 2379–2386. AAAI Press, 2019.

[Shperberg *et al.*, 2021] Shahaf S. Shperberg, Steven Danishevski, Ariel Felner, and Nathan R. Sturtevant. Iterative-deepening bidirectional heuristic search with restricted memory. In *ICAPS*, pages 331–339. AAAI Press, 2021.

[Sturtevant *et al.*, 2020] Nathan R. Sturtevant, Shahaf S. Shperberg, Ariel Felner, and Jingwei Chen. Predicting the effectiveness of bidirectional heuristic search. In *ICAPS*, pages 281–290. AAAI Press, 2020.

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