Bidirectional Bounded-Suboptimal Heuristic Search with Consistent Heuristics

Shahaf S. Shperberg[1], Natalie Morad[1], Lior Siag[1], Ariel Felner[1], Dor Atzmon[2]

[1] Faculty of Computer and Information Science, Ben-Gurion University of the Negev, Israel
[2] Department of Computer Science, Bar-Ilan University, Israel
{shperbsh,felner}@bgu.ac.il, {moradna,siagl}@post.bgu.ac.il, dor.atzmon@biu.ac.il

Abstract

Recent advancements in bidirectional heuristic search have yielded significant theoretical insights and novel algorithms. While most previous work has concentrated on optimal search methods, this paper focuses on bounded-suboptimal bidirectional search, where a bound on the suboptimality of the solution cost is specified. We build upon the state-of-the-art optimal bidirectional search algorithm, BAE*, designed for consistent heuristics, and introduce several variants of BAE* specifically tailored for the bounded-suboptimal context. Through experimental evaluation, we compare the performance of these new variants against other bounded-suboptimal bidirectional algorithms as well as the standard weighted A* algorithm. Our results demonstrate that each algorithm excels under distinct conditions, highlighting the strengths and weaknesses of each approach.

Code — https://github.com/SPL-BGU/Bounded-Suboptimal-BAE
Extended version — https://arxiv.org/abs/2511.10272

1 Introduction

In optimal search, the task is to find the least-cost path between two vertices, $start$ and $goal$, in a given graph. Admissible *Unidirectional Heuristic Search* (UHS) algorithms like A* (Hart, Nilsson, and Raphael 1968), which prioritize nodes by $f(n) = g(n) + h(n)$, return an optimal solution (Dechter and Pearl 1985) (of cost C^*) if h is admissible (never overestimating).

In many cases, finding optimal solutions is infeasible due to the immense computation required. *Bounded-suboptimal search* (BSS) is a paradigm that trades the quality of the solution for faster running time. In BSS, we are given a bound $W \geq 1$, and are required to find a solution with cost $\leq W \cdot C^*$. A classical BSS algorithm is *Weighted A** (WA*) (Pohl 1970), which expands nodes n according to $f(n) = g(n) + W \cdot h(n)$. Given an admissible heuristic, WA* is guaranteed to return a bounded-suboptimal solution. Other BSS algorithms include *Dynamic Potential Search* (DPS) (Gilon, Felner, and Stern 2016), Explicit Estimation Search (EES) (Thayer and Ruml 2011), XDP and XUP (Chen and Sturtevant 2019, 2021), and improved versions of these algorithms (Fickert, Gu, and Ruml 2022).

Bidirectional heuristic search (BiHS) is an alternative to UHS that progresses simultaneously from $start$ (forward) and $goal$ (backward) until the two frontiers meet. Recent line of work presented theoretical study and practical ways for using BiHS with tremendous achievements (Barker and Korf 2015; Holte et al. 2017; Eckerle et al. 2017; Shaham et al. 2017; Chen et al. 2017; Shperberg et al. 2019a,b, 2021; Alcázar, Riddle, and Barley 2020; Alcázar 2021; Siag et al. 2023b). Yet, while unidirectional BSS was broadly studied, bidirectional BSS has received limited attention.

BAE* (Sadhukhan 2013) (and the identical DIBBS algorithm (Sewell and Jacobson 2021)), is a BiHS optimal algorithm designed for consistent heuristics. BAE* prioritizes nodes in the open lists according to $b(n) = g(n) + h(n) + d(n)$ with $d(n)$ representing a lower bound on the heuristic error of paths discovered on the opposite search that passes through n. BAE* is considered the state-of-the-art optimal BiHS algorithm, with several studies highlighting its superiority (Alcázar, Riddle, and Barley 2020; Siag et al. 2023b).

In this work, we introduce a family of algorithms, called WBAE*, that adapt BAE* into the BSS paradigm by incorporating the basic idea of WA*, inflating h by W. Members of this family differ in how they treat d. We prove that multiplying d by any real number $\lambda \leq W$ results in a BSS solution. Additionally, we discuss two methods that achieve tighter lower bounds for WBAE* (as well as for other algorithms) and allow earlier termination, albeit sometimes at the cost of additional time overhead. We analyze the impact of λ and the tighter bounds, offering recommendations on how to set λ and choose the most effective termination condition.

2 Bidirectional Heuristic Search: Definitions

In BiHS, the aim is to find a least-cost path, of cost C^*, between $start$ and $goal$ in a given graph G. $c(x, y)$ denotes the cost of the cheapest path between x and y, so $c(start, goal) = C^*$. BiHS executes a forward search (F) from $start$ and a backward search (B) from $goal$ until the two searches meet. BiHS algorithms typically maintain two open lists OPEN_F and OPEN_B for the forward and backward searches, respectively. Each node has a g-value, an h-value, and an f-value (g_F, h_F, f_F and g_B, h_B, f_B for the forward and backward searches, correspondingly). For a direction D

Algorithm 1: BiHS General Framework

$U \leftarrow \infty$; $LB \leftarrow ComputeLowerBound()$;
while $\text{OPEN}_\text{F} \neq \emptyset \wedge \text{OPEN}_\text{B} \neq \emptyset \wedge U > LB$ **do**
  $D \leftarrow ChooseDirection()$;
  $n \leftarrow ChooseNode(D)$;
  $Expand(n, D)$; `// Also updates U`
  $LB \leftarrow ComputeLowerBound()$
return U

(F or B), f_D, g_D, and h_D represent the f-, g-, and h-values in that direction. We use $xMin_D$ to denote the minimal x value in OPEN_D, e.g., $gMin_F$ represents the minimal g-value in OPEN_F. Most BiHS algorithms consider the two *front-to-end* heuristic functions (Kaindl and Kainz 1997) $h_F(s)$ and $h_B(s)$ which respectively estimate $c(s, goal)$ and $c(start, s)$ for all $s \in G$. h_F is *forward admissible* iff $h_F(s) \leq c(s, goal)$ for all s in G and is *forward consistent* iff $h_F(s) \leq c(s, s') + h_F(s')$ for all s and s' in G. Backward *admissibility* and *consistency* are defined analogously.

BiHS algorithms differ primarily in their direction and node selection strategies, and in their termination criteria. A general BiHS framework is presented in Algorithm 1. U represents the cost of the incumbent solution and is returned if $U \leq LB$, where LB is a lower bound on the cost of the desired solution (either optimal or bounded suboptimal). During each expansion cycle, the algorithm chooses a direction D, then selects a node to expand from that direction. New nodes are matched against the opposite frontier, and if a better solution is found, U is updated. Subsequently, LB is updated as the content of OPEN_D has changed. BiHS algorithms vary in how they choose the direction, select nodes, and compute LB. A common direction choosing policy is the *alternate* policy, which chooses to expand from OPEN_F and OPEN_B in a round-robin fashion. Unless stated otherwise, all algorithms in this paper use this policy.

3 Background: BiHS Algorithms

We review several optimal and suboptimal BiHS algorithms.

3.1 WA* and Bidirectional WA*

A key BSS algorithm is weighted A* (WA*) (Pohl 1970), which replaces A*'s $f(n) = g(n) + h(n)$ (Hart, Nilsson, and Raphael 1968) priority function with $f_W(n) = g(n) + W \cdot h(n)$, inflating the heuristic by a user-defined $W \geq 1$. This guarantees a solution with cost of at most $W \cdot C^*$.

BHPA (Pohl 1971), or Bidirectional A* (BiA*), directly generalizes A* to BiHS. BiA* orders nodes in OPEN_D according to $f_D(n) = g_D(n) + h_D(n)$. Köll and Kaindl (1993) extend BiA* to BSS by applying the WA* approach to both search frontiers, a variant denoted here as Bidirectional WA* (WBiA*). WBiA* prioritizes nodes in OPEN_D according to:

$$pr_{W_D}(n) = g_D(n) + W \cdot h_D(n)$$

WBiA* detects solutions by locating the same state on both open lists and terminates once $U \leq LB_W$, where

$$LB_W = \max(pr_W Min_F, pr_W Min_B) \tag{1}$$

In this definition, $pr_W Min_F$ and $pr_W Min_B$ are the minimal pr_W-values in OPEN_F and OPEN_B, respectively. Köll and Kaindl (1993) did not provide proof for its suboptimality. For completeness, we provide it in Appendix A (available in the extended version).

3.2 Meet in the Middle (MM) and WMM

The *Meet in the middle* (MM) algorithm (Holte et al. 2017) is a notable BiHS that ensures that the search frontiers *meet in the middle* (i.e., no node n is expanded with $g(n) > C^*/2$). In MM, nodes n in OPEN_D are prioritized by:

$$pr_D(n) = \max(f_D(n), 2g_D(n))$$

MM expands the node with minimal priority, $PrMin$, among all nodes in OPEN_F and OPEN_B.

Weighted MM. Recently, MM was generalized to a BSS version called WMM (Atzmon et al. 2023). The main idea is to follow the WA* approach and inflate the heuristic by W. Several variants were proposed, and the best one prioritizes nodes in OPEN_D using the following formula:

$$pr_D(n) = g_D(n) + \max(g_D(n), W \cdot h_D(n))$$

WMM is *W-restrained*, i.e. the forward search never expands a node n with $g_F(n) > {}^{W}\!/_{2} \cdot C^*$ and the backward search never expands a node n with $g_B(n) > {}^{W}\!/_{2} \cdot C^*$.

3.3 BS* and Its BSS Variant

All previously discussed algorithms assume an *admissible* heuristic. In contrast, BS* (Kwa 1989) also requires *consistency* and improves upon BiA* by leveraging this property for tighter lower bounds and pruning via "nipping" and "trimming." It uses Pohl's *cardinality criterion* (Pohl 1971), expanding from the side with fewer open nodes. Köll and Kaindl (1993) proposed WBS*, a bounded-suboptimal variant of BS*. Like WBiA*, it uses $f_W(n) = g(n) + W \cdot h(n)$ and applies BS*'s enhancements to f_W instead of f.

3.4 BAE*

BAE* (Sadhukhan 2013) and the equivalent, DIBBS (Sewell and Jacobson 2021), are more recent BiHS algorithms that, like BS*, assume heuristic consistency. However, BAE* and DIBBS exploit this consistency more effectively than BS*, leading to improved search performance.

Let $d_F(n) = g_F(n) - h_B(n)$, the *difference* between the actual forward cost of n (from $start$) and its heuristic estimation to $start$. This indicates the *heuristic error* for node n (as $h_B(n)$ is a possibly inaccurate estimation of $g_F(n)$). Likewise, $d_B(m) = g_B(m) - h_F(m)$. BAE* orders nodes in OPEN_D according to:

$$b_D(n) = g_D(n) + h_D(n) + (g_D(n) - h_{\overline{D}}(n)) = f_D(n) + d_D(n)$$

where $h_{\overline{D}}$ is the heuristic in the direction opposite of D. $b_D(n)$ adds the heuristic error $d_D(n)$ to $f_D(n)$ to account for the underestimation by $h_{\overline{D}}(n)$. During each expansion cycle, BAE* chooses a search direction D and expands a node with minimal b_D-value. Additionally, BAE* detects solutions by locating the same state on both open lists and terminates once $U \leq LB_B$, where

$$LB_B = (bMin_F + bMin_B)/2 \tag{2}$$

in which $bMin_D$ is the minimal b-value in OPEN_D.

Given a consistent heuristic, BAE* was proven to return optimal solutions. Its informed priority function $b(n)$ helps it significantly outperform common UHS and BiHS algorithms, with improvements of up to an order of magnitude (Alcázar, Riddle, and Barley 2020; Siag et al. 2023a). Therefore, BAE* is considered a state-of-the-art BiHS algorithm for consistent heuristics. In this paper, we develop and study BSS variants of BAE*.

Other BSS BiHS algorithms have been proposed. A*-connect (Islam, Narayanan, and Likhachev 2016), designed for motion planning, extends BiA* using an additional inadmissible heuristic to quickly connect the frontiers. As our approach does not rely on such heuristics, A*-connect is not a direct competitor. Similarly, R2R (Rice and Tsotras 2012) is an *additive* BSS algorithm with guarantees of $C^* + W$, whereas we focus on *multiplicative* bounds ($W \cdot C^*$), making R2R incomparable in scope.

4 Weighted BAE*

We now introduce WBAE*, a BSS variant of BAE* that adopts the WA* idea of inflating h by a factor $W \geq 1$, guaranteeing a solution within $W \cdot C^*$. Recall the BAE* priority:

$$b_D(n) = g_D(n) + h_D(n) + d_D(n)$$

In WA*, the h-value is scaled by W to guide the search while preserving suboptimality bounds—a strategy we retain. The key challenge is handling the heuristic error $d_D(n)$, which has not been addressed in BSS settings.

We define the WBAE* priority function as:

$$b_{W_D}(n) = g_D(n) + W \cdot h_D(n) + \lambda \cdot d_D(n) \tag{3}$$

Here, $\lambda \leq W$ controls the impact of the error term. Special cases include: $\lambda = 0$, yielding a WBiA*-like function; $\lambda = W$, fully incorporating $d_F(n)$ into the heuristic; and intermediate values, which balance between the two. WBAE* detects solutions by locating the same state on both open lists and terminates once $U \leq LB_{WB}$, where

$$LB_{\texttt{WB}} = (b_W Min_F + b_W Min_B)/2 \tag{4}$$

and $b_W Min_D$ is the minimal b_{W_D}-value in OPEN_D.

Notably, BAE* guarantees optimality only when given consistent heuristics, due to the nature of the error-correction term d_D. Consequently, WBAE* requires heuristic consistency to ensure bounded suboptimality. Throughout the remainder of the paper, we therefore assume that the given heuristic h is consistent.

4.1 Theoretical Analysis

We first show that WBAE* is bounded-suboptimal. We then discuss the role of λ and provide guidelines for selecting values for λ. To prove that WBAE* is bounded-suboptimal, we start by proving the following lemma.

Lemma 1. *For every node n and any $\lambda \leq W$, it holds that $b_{W_D}(n) \leq W \cdot b_D(n)$.*

Proof. Let n be a node discovered during the search in direction D. We want to show that $b_{W_D}(n) \leq W \cdot b_D(n)$. By using the definitions of b_{W_D} and b_D, we get:

$$b_{W_D}(n) = g_D(n) + W \cdot h_D(n) + \lambda \cdot (g_D(n) - h_{\overline{D}}(n))$$

and similarly,

$$W \cdot b_D(n) = W \cdot (g_D(n) + h_D(n) + (g_D(n) - h_{\overline{D}}(n)))$$

Substituting these into the inequality in Lemma 1, $b_{W_D}(n) \leq W \cdot b_D(n)$, we now need to prove that:

$$g_D(n) + W \cdot h_D(n) + \lambda \cdot (g_D(n) - h_{\overline{D}}(n)) \leq$$
$$W \cdot g_D(n) + W \cdot h_D(n) + W \cdot (g_D(n) - h_{\overline{D}}(n))$$

By eliminating $W \cdot h_D(n)$ from both sides and simplifying, the inequality that we need to prove becomes:

$$(W - \lambda)h_{\overline{D}}(n) \leq (2W - \lambda - 1)g_D(n) \tag{5}$$

Since $W \geq \lambda$ and $W \geq 1$ by definition, it follows that $(2W - \lambda - 1) \geq 0$. Additionally, because the heuristic is admissible, we have $0 \leq h_{\overline{D}}(n) \leq g_D(n)$. Consequently, $(2W - \lambda - 1)h_{\overline{D}}(n) \leq (2W - \lambda - 1)g_D(n)$. To prove Inequality 5, it is therefore sufficient to demonstrate that:

$$(W - \lambda)h_{\overline{D}}(n) \leq (2W - \lambda - 1)h_{\overline{D}}(n) \tag{6}$$

By simplifying the inequality, we get:

$$(1 - W)h_{\overline{D}}(n) \leq 0 \tag{7}$$

For $h_{\overline{D}}(n) = 0$, inequality 7 holds trivially. For $h_{\overline{D}}(n) > 0$, the inequality holds when $W \geq 1$, which holds by definition. Therefore, the inequality holds and $b_{W_D}(n) \leq W \cdot b_D(n)$ for all nodes n. □

We proceed by proving the bounded-suboptimality of WBAE* by making use of Lemma 1.

Theorem 2. *WBAE* is bounded-suboptimal.*

Proof. Previous analysis of BAE* (Sadhukhan 2013) shows that when the heuristic function is consistent, it holds that $\frac{bMin_F + bMin_B}{2} \leq C^*$ throughout the search. Consequently, it is sufficient to show that throughout the search, $\frac{b_W Min_F + b_W Min_B}{2} \leq W \cdot \frac{bMin_F + bMin_B}{2}$, as this would imply that $\frac{b_W Min_F + b_W Min_B}{2} \leq W \cdot C^*$.

Let n_f and n_b be the nodes with the minimal b_F ($bMin_F$) and b_B ($bMin_B$) values, respectively, at some arbitrary iteration during the search. From Lemma 1, we know that $b_{W_F}(n_f) \leq W \cdot b_F(n_f)$ and $b_{W_B}(n_b) \leq W \cdot b_B(n_b)$. Thus,

$$\begin{aligned} \frac{b_W Min_F + b_W Min_B}{2} &\leq \frac{b_{W_F}(n_f) + b_{W_B}(n_b)}{2} \\ &\leq W \cdot \frac{b_F(n_f) + b_B(n_b)}{2} \\ &= W \cdot \frac{bMin_F + bMin_B}{2} \leq W \cdot C^*. \end{aligned}$$

Since WBAE* terminates only when a solution is found with cost $\leq \frac{b_W Min_F + b_W Min_B}{2}$, it is guaranteed to return a solution with a cost $\leq W \cdot C^*$ (BSS) This ensures that WBAE* is a bounded-suboptimal algorithm. □

Finally, we show that similar to WA*, WBAE* can avoid node re-expansions when given consistent heuristics.

Theorem 3. *Given consistent heuristics, WBAE* finds a bounded-suboptimal solution without re-expanding nodes.*

Proof. Due to the priority function of WBAE*, every node n in direction D that undergoes expansion is ensured to have been reached through a bounded suboptimal path, specifically satisfying, $g_D(n) \leq W \cdot c(start, n)$. For a more detailed proof, refer to Appendix B. □

4.2 The Role of λ

In this section, we discuss the role of λ and provide guidelines for selecting an appropriate value for it.

In heuristic search, algorithms must perform two parallel tasks. The first task is to find and refine solutions, which corresponds to lowering the upper bound (U). The second task is proving the (sub)optimality of solutions, corresponding to increasing the lower bound (LB). Search algorithms can only terminate when $U \leq LB$, making both tasks essential. However, the approaches for tackling each task often contradict one another. For example, the DVCBS algorithm (Shperberg et al. 2019b) excels at increasing LB but often finds solutions slower than other algorithms, whereas Greedy Best-First Search (Doran and Michie 1966), which prioritizes nodes solely by their h-values, is good at finding solutions, but is not concerned with proving optimality.

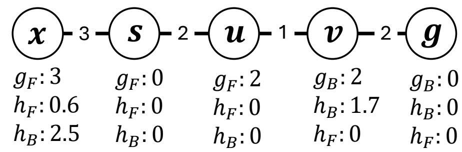
We next show that while incorporating the heuristic error d supports the task of proving solution suboptimality (the second task), it can hinder the process of finding a solution (the first task). Given two nodes, n_1 and n_2, in direction D with $g_D(n_1)+W\cdot h_D(n_1) = g_D(n_2)+W\cdot h_D(n_2)$, WBAE* with $\lambda > 0$ delays the expansion of the node with the higher heuristic error d. By definition, $d(n) = g_D(n)-h_{\overline{D}}(n)$. Due to admissibility, $g_D(n) \geq h_{\overline{D}}(n) \geq 0$. Thus, for every node n, $d(n) \leq g_D(n)$. Consequently, nodes with higher g-values tend to have higher d values, and their expansion is often delayed by WBAE*. Expanding nodes with high g-values is essential for finding solutions. Therefore, WBAE* is expected to be less efficient than WBiA* in finding solutions. Nonetheless, d offers a more accurate estimate of the lower bound on solution cost, making a larger λ more effective for establishing the (sub)optimality of solutions.

Due to the trade-off between increasing LB and decreasing U, different approaches are required based on the specific challenge of each task. At one extreme, if the initial heuristic yields a sufficiently high lower bound to guarantee (sub)optimality upon finding a solution, the focus should be on finding a solution—favoring a small (or zero) λ, especially when valid solutions are rare. Conversely, when many (sub)optimal solutions exist but proving their quality requires extensive search, the emphasis should be on raising LB, implying a large λ. In general, we expect the task of increasing LB to be dominant when: (i) the heuristic is weak, significantly underestimating the shortest paths from many states, and (ii) W is small, as the effort to prove suboptimality increases. Thus, we propose the following hypothesis:

Hypothesis on the role of λ: Larger values of λ are more effective when the heuristic is weak and the suboptimality bound W is small. Smaller values of λ are preferable when the heuristic is more accurate or when W is large.

Figure 1 presents a small example illustrating how different values of λ affect node expansions during search. The top displays a problem instance with heuristic values and edge costs. We consider WBAE* with $W = 1.1$ and $W = 5$, and two λ-values, $\lambda = W$ and $\lambda = W^{-2}$; priorities for each configuration are given inside the table (excluding nodes s and g, which are expanded first regardless).

All algorithms first expand s and g. Now, consider $w = 1.1$. With both values of λ, u is expanded, yielding a solution of cost 5. For $\lambda = W$, the search can terminate: the lowest priority in the backward direction is 6.07, and in the forward direction, it is at most the priority of x, which is 4.21. Hence, the lower bound $(6.07 + 4.21)/2 = 5.14$ exceeds 5. In contrast, with $\lambda = W^{-2}$, after u is expanded, the minimal forward priority is at most that of x, giving a lower bound of $(5.52 + 4.07)/2 = 4.795$, which is less than 5, requiring the expansion of x. For $w = 5$ and $\lambda = W^{-2}$, u is expanded and the search terminates. With $\lambda = W$, x is expanded even before u, leading to more node expansions.



State	Dir.	$W = 1.1$		$W = 5$	
		$\lambda = W$	$\lambda = W^{-2}$	$\lambda = W$	$\lambda = W^{-2}$
u	F	4.20	3.65	12.0	2.08
x	F	4.21	4.07	8.5	6.02
v	B	6.07	5.52	20.5	10.58

Figure 1: An example demonstrating the role of λ

This example supports our hypothesis above. For small values of w ($w = 1.1$) larger values of λ ($\lambda = W$) perform better than smaller values ($\lambda = W^{-2}$). For large values of W ($W = 5$) small values of λ ($\lambda = W^{-2}$) perform better than larger values ($\lambda = W$).

5 Obtaining Stronger Lower Bounds

In this section, we examine two methods to tighten lower bounds throughout the search, potentially reducing the effort needed by BSS algorithms to return a suboptimal solution.

5.1 Alternative Termination Criteria

Algorithms in the WA* family typically terminate based on the minimum values of their priority, which multiply the heuristic h (but not the cost g) by W. For example, WBiA* terminates when $U \leq \max(pr_W Min_F, pr_W Min_B)$, where U is the cost of the incumbent solution, and $pr_W Min_F$, $pr_W Min_B$ are the minimal pr_W-values in the open lists.

Nonetheless, $fMin_F$ and $fMin_B$ are still lower bounds on C^*. Consequently, $W\cdot\max(fMin_F, fMin_B) \leq W\cdot C^*$, and therefore $U \leq W\cdot\max(fMin_F, fMin_B)$ is an alternative termination condition that still maintains the bounded-suboptimality guarantees. Moreover, let n be the node with the minimal f-value in direction D. It holds that,

$$W\cdot fMin_D = W\cdot g_D(n) + W\cdot h_D(n) \geq g_D(n) + W\cdot h_D(n) \geq pr_{W_D}(n) \geq pr_W Min_D$$

This means that the alternative lower bound (denoted ALB) is tighter than the original lower bound, which can result in fewer expansions during the search.

The above ALB was defined based on $fMin$. Nonetheless, a similar ALB can be defined for all algorithms in the WA* family by taking the bounds of their corresponding

Algorithm	ToH	DAO	Mazes	Pancake	STP
WA*	697K	1,585K	2,024K	56K	524K
WBiA*	601K	1,466K	1,694K	76K	502K
WMM	589K	845K	606K	75K	400K
WBS*	644K	1,564K	1,985K	67K	497K
WBAE* $\frac{1}{W^2}$	607K	1,447K	1,603K	66K	465K
WBAE* $\frac{1}{W}$	540K	1,447K	1,646K	66K	450K
WBAE* 1	534K	1,468K	1,717K	65K	428K
WBAE* W	523K	1,464K	1,667K	61K	336K
WBAE* λ	478K	1,454K	1,709K	60K	430K

Table 1: Expansions/sec averaged across all weights W.

optimal variant and multiplying them by W. For WBAE*, ALB will therefore halt the search when $U \leq W \cdot LB_B$.

Nevertheless, using ALB incurs extra overhead as it maintains both priorities (e.g., $pr_W Min_D$ and $fMin_D$ for WBiA*, or $b_W Min_D$ and $bMin_D$ for WBAE*). This might require storing two open lists for each frontier. We analyze the improvement resulting from ALB, as well as the computational overhead, in our empirical evaluation below.

5.2 Utilizing Information on Edge Costs

Alcázar, Riddle, and Barley (2020) demonstrated how edge cost information can tighten lower bounds on C^*throughout the search. They assumed the algorithm knows the *greatest common denominator* (GCD) of all edge costs, ι, and used it to redefine BAE*'s termination condition:

$$LB_B = \left\lceil \frac{(bMin_F + bMin_B)/2}{\iota} \right\rceil \cdot \iota$$

this is a lower bound on C^*, as solutions can only increase in GCD increments. Thus, if a lower bound suggests a solution not divisible by ι, it can be rounded up to the nearest ι increment. For example, if $\iota = 3$ and the lower-bound $LB = 14$, $\lceil 14/3 \rceil \cdot 3 = 15$ can be used as a tighter lower bound. A similar approach could apply to A*'s lower bound, but typically, $fMin = g(n) + h(n)$ is already divisible by ι since $g(n)$ is always divisible by ι, and $h(n)$ is often as well.

This alteration to the lower bound formula can also be applied to the BSS variants of BAE* as follows:

$$LB_{\mathtt{WB}} = \left\lceil \frac{(b_W Min_F + b_W Min_B)/2}{\iota \cdot W} \right\rceil \cdot \iota \cdot W$$

Let n_f and n_b be the nodes with the minimal b-value in OPEN$_\text{F}$ and OPEN$_\text{B}$, respectively, at some arbitrary iteration during the search. This formulation of $LB_{\mathtt{WB}}$ still yields bounded solutions, since:

$$\begin{aligned}\left\lceil \frac{(b_W Min_F + b_W Min_B)/2}{\iota \cdot W} \right\rceil \cdot \iota \cdot W &\leq \left\lceil \frac{(b_{W_F}(n_f) + b_{W_F}(n_B))/2}{\iota \cdot W} \right\rceil \cdot \iota \cdot W \\ &\leq \left\lceil \frac{(W \cdot (b_F(n_f) + b_F(n_B))/2}{\iota \cdot W} \right\rceil \cdot \iota \cdot W \\ &= \left\lceil \frac{W \cdot (bMin_F + bMin_B)/2}{\iota \cdot W} \right\rceil \cdot \iota \cdot W \\ &= \left\lceil \frac{(bMin_F + bMin_B)/2}{\iota} \right\rceil \cdot \iota \cdot W \leq C^* \cdot W\end{aligned}$$

Similar adjustments can be applied to all WA*-family algorithms. Notably, for $W > 1$, this revised formulation can benefit WBiA* (and also WA*), as $fMin = g(n) + W \cdot h(n)$ is not guaranteed to be divisible by W. Moreover, unlike ALB, which requires maintaining multiple open lists that incur additional computational overhead, the GCD improvement adds no computational cost.

6 Empirical Evaluation

We now present an empirical evaluation comparing BiHS BSS algorithms, including our newly introduced WBAE*. This evaluation also tests our hypothesis on the role of λ and assesses the impact of the alternative termination conditions.

Baselines. We compare all existing BiHS BSS algorithms: WBiA*, WBS*, WMM, and WBAE*, as well as WA* as a representative UniHS BSS algorithm. For WBAE*, we evaluate several values of $\lambda \in \{0, \frac{1}{W^2}, \frac{1}{W}, 1, W\}$ to examine its impact on performance. Since WBAE* with $\lambda = 0$ uses the same priority rule as WBiA*, we omit it from the results. All algorithms break ties in favor of higher g-values.

In addition, we report results for a *tuned* λ value, denoted λ^*, selected separately for each domain and each W. To determine λ^*, we used the Optuna (Akiba et al. 2019) hyperparameter optimization framework. Optuna performed 50 trials; in each, it selected a λ value and ran WBAE* on an additional set of 100 domain-specific instances disjoint from the test set. The value with the best overall tuning performance was used for testing (listed in Appendix F).

Domains. We experimented on five domains: **(1)** 100 instances of the 12-disk *4-peg Towers of Hanoi* problem (ToH) with (10+2), (8+4), and (6+6) additive Pattern Databases (PDBs) (Felner et al. 2004). **(2)** The standard 100 instances of the *15 puzzle* problem (STP) (Korf 1985) using the Manhattan distance heuristic. **(3)** 100 instances of the 18-pancake puzzle with the GAP heuristic (Helmert 2010). To explore different heuristic strengths, we also used GAP-n heuristics (for $n = 1, 2$) where the n smallest pancakes are excluded from the heuristic computation. We also experimented on the following *Grid*-based pathfinding maps from the *MovingAI* repository (Sturtevant 2012) using octile distance as a heuristic and 1.5 cost for diagonal edges: **(4)** 156 maps from Dragon Age Origins (DAO), each with different start and goal points (totaling 3149 instances), and **(5)** mazes with varying widths (1200 instances). Heuristic values for states were set to the maximum of the heuristic function's estimate and 1, the minimum edge cost (commonly referred to as ϵ). Throughout the evaluation, we explore various values of W ranging from 1 to 10, specifically $W \in \{1, 1.1, 1.2, 1.5, 1.7, 2, 3, 5, 10\}$.

Runtime. The different priority functions and the GCD enhancement had a small impact on the constant time per node. Table 1 presents the average node expansions per second for each algorithm (including the GCD enhancement) across various domains. Although there are some variations between domains, the overall overhead of node expansions is comparable, and the trends observed below in node expansions are largely preserved when considering runtime as well. Comprehensive runtime results, broken down by heuristic and weight, can be found in Appendix C.

The ALB enhancement adds computational overhead and is not always time-efficient. Thus, the initial experiments include the GCD improvement but exclude ALB, as this combination was generally the most efficient. The impact of both GCD and ALB—especially on CPU time—is analyzed later.

ToH Results. Table 2 shows the average number of node expansions in the ToH domain using (10+2), (8+4), and

Algorithm	ToH-12 (10+2) 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	279K	137K	54K	4K	2K	1K	417	433	441
WBiA*	236K	106K	40K	**3K**	**1K**	**627**	**374**	**343**	347
WMM	337K	164K	60K	3K	2K	2K	5K	4K	556
WBS*	179K	82K	33K	**3K**	**1K**	649	377	349	353
WBAE* $\frac{1}{W^2}$	**48K**	30K	**17K**	5K	3K	1K	394	353	343
WBAE* $\frac{1}{W}$	**48K**	**28K**	18K	7K	4K	1K	405	364	343
WBAE* 1	**48K**	**28K**	20K	9K	6K	3K	536	380	357
WBAE* W	**48K**	29K	23K	13K	11K	7K	3K	2K	1K
WBAE* λ^*	**48K**	**28K**	**17K**	**3K**	2K	789	394	356	**342**

Algorithm	ToH-12 (8+4) 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	2M	2M	1M	486K	318K	278K	26K	13K	20K
WBiA*	1M	806K	558K	186K	109K	39K	**11K**	**5K**	**4K**
WMM	1M	863K	707K	269K	157K	56K	15K	10K	8K
WBS*	924K	656K	466K	167K	94K	35K	**11K**	6K	**4K**
WBAE* $\frac{1}{W^2}$	**189K**	182K	162K	105K	74K	34K	12K	**5K**	**4K**
WBAE* $\frac{1}{W}$	**189K**	162K	137K	**97K**	76K	42K	14K	6K	**4K**
WBAE* 1	**189K**	148K	**130K**	105K	92K	69K	22K	9K	5K
WBAE* W	**189K**	**143K**	131K	109K	102K	88K	71K	56K	41K
WBAE* λ^*	191K	144K	132K	98K	**72K**	**32K**	**11K**	**5K**	**4K**

Algorithm	ToH-12 (6+6) 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	3M	3M	3M	2M	2M	1M	1M	863K	331K
WBiA*	2M	2M	1M	682K	439K	266K	117K	72K	**43K**
WMM	964K	1M	1M	842K	600K	395K	251K	121K	64K
WBS*	2M	1M	1M	584K	390K	248K	**115K**	76K	49K
WBAE* $\frac{1}{W^2}$	**382K**	403K	402K	358K	320K	**221K**	119K	**71K**	45K
WBAE* $\frac{1}{W}$	**382K**	362K	342K	310K	284K	227K	130K	77K	47K
WBAE* 1	**382K**	333K	317K	301K	292K	263K	176K	111K	68K
WBAE* W	**382K**	**318K**	**306K**	**282K**	**273K**	252K	227K	202K	186K
WBAE* λ^*	386K	319K	309K	309K	284K	223K	121K	**71K**	45K

Table 2: Average number of node expansions on the 12-disks ToH domain, with $(10+2)$, $(8+4)$, and $(6+6)$ PDBs

Algorithm	STP 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	16M	10M	3M	320K	103K	41K	12K	**5K**	4K
WBiA*	19M	4M	658K	78K	50K	**29K**	**11K**	**5K**	4K
WMM	15M	4M	861K	162K	160K	270K	2M	3M	47K
WBS*	12M	3M	486K	**76K**	**49K**	**29K**	**11K**	**5K**	4K
WBAE* $\frac{1}{W^2}$	**3M**	**1M**	449K	143K	80K	34K	12K	**5K**	**3K**
WBAE* $\frac{1}{W}$	**3M**	**1M**	586K	202K	139K	57K	16K	6K	**3K**
WBAE* 1	**3M**	**1M**	763K	337K	250K	171K	40K	11K	5K
WBAE* W	**3M**	**1M**	950K	688K	617K	553K	1M	831K	575K
WBAE* λ^*	**3M**	**1M**	**368K**	89K	60K	33K	12K	**5K**	**3K**

Algorithm	Mazes 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	99K	98K	96K	92K	90K	86K	76K	65K	**56K**
WBiA*	100K	99K	98K	95K	92K	89K	77K	66K	59K
WMM	85K	85K	85K	84K	84K	84K	80K	70K	60K
WBS*	88K	87K	86K	83K	82K	78K	**69K**	**60K**	**56K**
WBAE* $\frac{1}{W^2}$	**81K**	85K	88K	90K	90K	87K	77K	66K	59K
WBAE* $\frac{1}{W}$	**81K**	83K	84K	85K	84K	83K	76K	66K	59K
WBAE* 1	**81K**	**80K**	**80K**	**79K**	**78K**	**77K**	74K	69K	62K
WBAE* W	**81K**	**80K**	**80K**	80K	79K	79K	78K	77K	77K
WBAE* λ^*	81K	**80K**	**80K**	**79K**	**78K**	**77K**	73K	66K	59K

Algorithm	DAO 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	**5K**	**5K**	**4K**	**4K**	**3K**	**3K**	**2K**	**2K**	**2K**
WBiA*	7K	6K	5K	**4K**	4K	**3K**	3K	3K	**2K**
WMM	8K	7K	6K	5K	5K	4K	3K	3K	**2K**
WBS*	6K	**5K**	5K	**4K**	4K	**3K**	3K	3K	3K
WBAE* $\frac{1}{W^2}$	7K	6K	6K	5K	4K	4K	3K	3K	**2K**
WBAE* $\frac{1}{W}$	7K	6K	6K	5K	4K	4K	3K	3K	**2K**
WBAE* 1	7K	6K	6K	5K	5K	4K	3K	3K	**2K**
WBAE* W	7K	6K	6K	5K	5K	5K	4K	4K	4K
WBAE* λ^*	7K	6K	5K	**4K**	4K	**3K**	3K	3K	**2K**

Table 3: Average number of node expansions on the STP, Mazes, and DAO domains

(6+6) additive PDB heuristics. In ToH, for optimal search ($W = 1$), WBAE* (effectively BAE*) significantly outperforms the other algorithms, and in particular, it outperforms WA* (effectively A*) by a factor of 6 to 10, depending on the heuristic. This demonstrates the importance of using the heuristic error d, when available, for proving the optimality of solutions. However, in accordance with our hypothesis, as W increases, the focus of the search shifts from proving optimality to finding a solution. Thus, the heuristic error d becomes less important, and can even hinder the search. For example, with the (10+2) PDB, $\lambda = W$ is never the best policy for $W > 1$ among all the examined values. For $W = 1.1$, we observe that $\lambda = 1$ and $\lambda = \frac{1}{W}$ have roughly the same performance (28K expansions), but for $W = 1.2$, a lower value of $\lambda = \frac{1}{W^2}$ yields the best results. For $W \geq 1.5$, it is better to ignore d altogether and use $\lambda = 0$ (= WBiA*).

For the weaker heuristics, (8+4) and (6+6), the trends are similar, although the transition point between the best values of λ is different. For example, with the (8+4) heuristic, $\lambda = W$ is best performing up to $W = 1.1$, while with the (6+6) heuristic, $\lambda = W$ is better up to $W = 2$. This observation is also in agreement with our hypothesis, as we see that when the heuristic gets weaker, higher values of λ tend to work better. Notably, in this domain, WA* was never the best choice, although the advantage of the BiHS algorithm diminishes as W increases. Moreover, neither WMM nor WBS* emerged as the best algorithms. WBiA* (similar to WBAE* with $\lambda = 0$) outperformed all other variants in some configurations with larger values of W, particularly when using the stronger heuristic (10+2). However, it struggled with smaller values of W, especially when combined with weaker heuristics (8+4 and 6+6). This pattern aligns with our hypothesis: when heuristics are strong and W is large, less effort is needed to validate (sub)optimality, diminishing the impact of λ. Notably, the tuned λ values, λ^*, achieved the best average performance across weights, often matching or outperforming the best fixed λ configurations.

STP, Mazes, and DAO. Table 3 shows the results for STP, Mazes, and DAO. For $W = 1$, WBAE* outperforms WA* (and all other algorithms) by a factor of 5. However, as W increases, the importance of d diminishes, and lower λ values become more effective. Notably, a small value of $\lambda = \frac{1}{W^2}$ performs well even for small W, due to the relatively strong MD heuristic. To test this effect with a weaker heuristic, we repeated the experiment using MD-4 (ignoring four central tiles, like gap-4). Full results are in Appendix E. Here, larger λ values performed better; for instance, with MD-4 and $W = 1.2$, $\lambda = 1$ led to fewer expansions than $\lambda = \frac{1}{W^2}$, supporting our hypothesis.

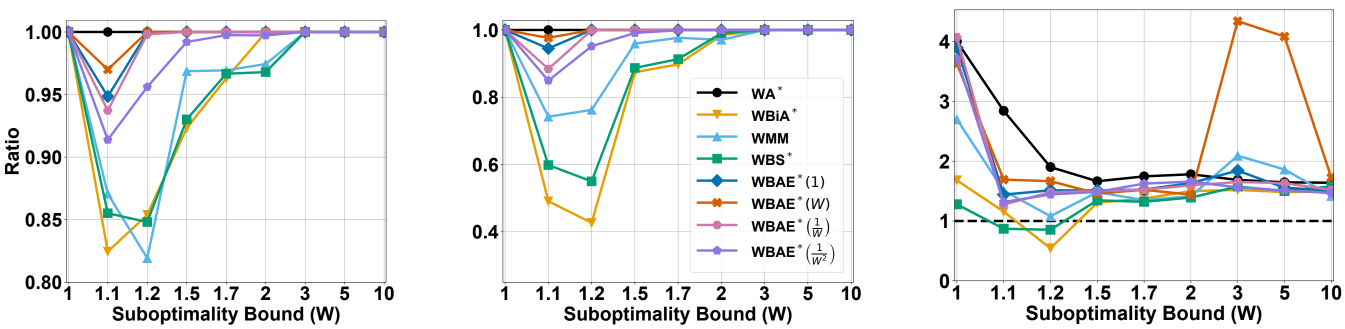
For $W \geq 1.5$, WBiA*—which does not use d—becomes the most effective algorithm, until $W = 10$, where $\lambda = \frac{1}{W}$ and $\lambda = \frac{1}{W^2}$ slightly outperform it. We also evaluated the heavy STP variant, a non-unit-cost domain where tile movement cost equals the tile's label. Results (see Appendix D) follow similar trends; however, in this case, WBAE* maintains a clear advantage over WA* even at large W, reducing node expansions by a factor of 5 when $W = 10$.

For Mazes and DAO, *all* algorithms perform similarly, especially as W increases. This is likely due to their polynomial state space, which diminishes algorithmic differences, unlike in (the other) exponential domains. Nonetheless, the λ trend (supporting our hypothesis) persists in Mazes: larger λ (W, 1) are better at $W = 1.1$, while smaller λ ($\frac{1}{W^2}$, 0) excel at $W = 5$ and $W = 10$.

In these domains as well, the tuned values λ^* consistently yielded strong performance, often matching or exceeding

Algorithm	GAP 1	1.1	1.2	1.5	1.7	2	3	5	10	GAP-1 1	1.1	1.2	1.5	1.7	2	3	5	10	GAP-2 1	1.1	1.2	1.5	1.7	2	3	5	10
WA*	**194**	147	**51**	**28**	**26**	**23**	**23**	**23**	**23**	42K	25K	3K	271	146	62	46	**42**	**42**	5M	5M	1M	77K	21K	4K	1K	1K	1K
WBiA*	225	**100**	60	41	40	37	37	37	37	61K	5K	893	59	49	**43**	**42**	**42**	**42**	8M	2M	194K	5K	1K	349	180	**164**	**167**
WMM	417	245	312	2K	5K	19K	5K	55	37	43K	4K	1K	2K	4K	10K	103K	665	54	3M	510K	90K	12K	5K	16K	134K	46K	437
WBS*	203	106	62	41	40	37	37	37	37	33K	4K	884	**51**	**48**	**43**	**42**	**42**	**42**	4M	1M	126K	4K	**527**	**254**	**175**	**164**	**167**
WBAE* $\frac{1}{W^2}$	205	111	78	43	41	37	36	36	36	**2K**	**596**	386	75	59	48	43	**42**	**42**	145K	48K	9K	3K	1K	540	180	176	180
WBAE* $\frac{1}{W}$	205	114	80	46	42	37	36	36	36	**2K**	637	511	120	71	50	43	**42**	**42**	145K	38K	**8K**	7K	2K	818	176	176	180
WBAE* 1	205	116	80	48	43	38	36	36	36	**2K**	687	580	164	117	60	45	**42**	**42**	145K	37K	11K	12K	9K	2K	336	197	192
WBAE* W	205	115	83	57	55	44	47	55	47	**2K**	742	873	2K	1K	845	385	216	178	145K	37K	17K	71K	132K	393K	85K	44K	26K
WBAE* λ^*	205	114	66	42	40	37	36	36	36	**2K**	583	**283**	75	60	50	43	**42**	**42**	**134K**	**35K**	**8K**	**2K**	1K	540	196	172	201

Table 4: Average number of node expansions on 18 Pancakes domain, with GAP to GAP-2 heuristics



(a) GCD Reduction in Expansions (b) ALB Reduction in Expansions (c) ALB Reduction in Runtime

Figure 2: Comparison of the methods for strengthening the lower bound on STP

the best fixed λ configurations in terms of average results.

18-pancake. Table 4 presents the results for the 18-pancake. Here, WA* shows superior performance for GAP, where the heuristic is highly accurate, as well as for large values of W across all heuristics. Nevertheless, the observed trend in the choice of λ remains consistent: the best-performing value of λ tends to decrease as heuristic accuracy increases and as W becomes larger. Specifically, for GAP and GAP-1, smaller values such as $\lambda = \frac{1}{W^2}$ and $\lambda = 0$ perform as well as or better than all other λ settings across all values of W. In contrast, for GAP-2, a weaker heuristic, larger values, such as $\lambda = W$ and $\lambda = 1$, are more effective when W is small.

Solution Quality. All algorithms produced solutions well within bounds. Even for $W = 10$, average costs never exceeded $2.3 \cdot C^*$. Full results are available in Appendix C.

6.1 Analysis of the Stronger Lower Bounds

Figures 2a and 2b illustrate the ratio in node expansions achieved by GCD and ALB, respectively, relative to a variant that does not use them (the 1.0 line). Additionally, the impact of ALB on the overall runtime is presented in Figure 2c. These are representative results for the 15-puzzle (15-STP); other domains showed similar trends. Notably, the GCD improvement imposes minimal computational overhead, so the reduction in node expansions directly corresponds to a decrease in runtime. Both methods show the most significant reduction in nodes for small values of W. For example, for $W \in \{1.1, 1.2\}$, GCD expanded approximately 85% of the nodes for WBiA*, WBS*, and WMM (again, at no extra CPU cost). ALB at $W = 1.2$ achieves a reduction to 40% nodes for WBiA* and almost 50% for WBS*. However, the reduction is minor for other algorithms and becomes negligible when $W > 3$. This trend highlights the diminishing complexity of proving solution optimality as W increases. Importantly, ALB only improves the overall runtime (indicated by being below the 1.0 line in Figure 2c) for $W \in \{1.1, 1.2\}$ for WBS* and when $W = 1.2$ for WBiA*; in all other scenarios, it adversely affects performance.

7 Summary and Conclusions

We integrated the core concept of Weighted A* (WA*) into BAE*, showing how the choice of λ, which weights the heuristic error d, affects performance. We proposed tighter lower bound methods and showed that with a well-chosen λ, WBAE* outperforms related algorithms in many cases. We also provided practical guidelines for using WBAE* and selecting λ, and conducted experiments with tuned λ values.

Future work could develop BiHS and BSS algorithms inspired by strategies beyond WA*, such as DPS (Gilon, Felner, and Stern 2016), EES (Thayer and Ruml 2011), and XDP/XUP (Chen and Sturtevant 2021). WBAE* could also be extended to use separate λ values for forward and backward search, which may be beneficial with asymmetric heuristics.

Acknowledgements

The work of Shahaf Shperberg and Ariel Felner was supported by the Israel Science Foundation (ISF) grant #909/23. This work was also supported by Israel's Ministry of Innovation, Science and Technology (MOST) grant #1001706842, in collaboration with Israel National Road Safety Authority and Netivei Israel, awarded to Shahaf Shperberg, and by BSF grant #2024614 awarded to Shahaf Shperberg. The work of Dor Atzmon was supported by the Israel Science Foundation (ISF) grant #1511/25. This work was also supported by Israel's Ministry of Innovation, Science and Technology (MOST) grant #6908 (Czech-Israeli cooperative scientific research) awarded to Dor Atzmon.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *KDD*, 2623–2631. ACM.

Alcázar, V. 2021. 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*, 7–15.

Alcázar, V.; Riddle, P. J.; and Barley, M. 2020. A Unifying View on Individual Bounds and Heuristic Inaccuracies in Bidirectional Search. In *AAAI*, 2327–2334.

Atzmon, D.; Shperberg, S. S.; Sabah, N.; Felner, A.; and Sturtevant, N. R. 2023. W-restrained Bidirectional Bounded-Suboptimal Heuristic Search. In *ICAPS*, 26–30.

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

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions. In *IJCAI*, 489–495.

Chen, J.; and Sturtevant, N. R. 2019. Conditions for Avoiding Node Re-Expansions in Bounded Suboptimal Search. In *IJCAI*, 1220–1226.

Chen, J.; and Sturtevant, N. R. 2021. Necessary and Sufficient Conditions for Avoiding Reopenings in Best First Suboptimal Search with General Bounding Functions. In *AAAI*, 3688–3696.

Dechter, R.; and Pearl, J. 1985. Generalized Best-First Search Strategies and the Optimality of A*. *JACM*, 32(3): 505–536.

Doran, J. E.; and Michie, D. 1966. Experiments with the graph traverser program. In *Proceedings of the Royal Society A*, volume 294, 235–259.

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*, 79–87.

Felner, A.; Korf, R. E.; Meshulam, R.; and Holte, R. C. 2004. Compressing Pattern Databases. In *AAAI*, 638–643.

Fickert, M.; Gu, T.; and Ruml, W. 2022. New Results in Bounded-Suboptimal Search. In *AAAI*, 10166–10173.

Gilon, D.; Felner, A.; and Stern, R. 2016. Dynamic Potential Search—A New Bounded Suboptimal Search. In *SoCS*, 36–44.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2): 100–107.

Helmert, M. 2010. Landmark Heuristics for the Pancake Problem. In *SoCS*, 109–110. AAAI Press.

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.

Islam, F.; Narayanan, V.; and Likhachev, M. 2016. A-Connect: Bounded suboptimal bidirectional heuristic search. In *ICRA*, 2752–2758.

Kaindl, H.; and Kainz, G. 1997. Bidirectional Heuristic Search Reconsidered. *J. Artif. Intell. Res.*, 7: 283–317.

Korf, R. E. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artif. Intell.*, 27(1): 97–109.

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

Köll, A. L.; and Kaindl, H. 1993. Bidirectional Best-First Search with Bounded Error: Summary of Results. In *IJCAI*, 217–223.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1: 193–204.

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

Rice, M. N.; and Tsotras, V. J. 2012. Bidirectional A* Search with Additive Approximation Bounds. In *SoCS*, 80–87. AAAI Press.

Sadhukhan, S. K. 2013. Bidirectional heuristic search based on error estimate. *CSI Journal of Computing*, 2(1-2): S1:57–S1:64.

Sewell, E. C.; and Jacobson, S. H. 2021. Dynamically improved bounds bidirectional search. *Artif. Intell.*, 291: 103405.

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.

Shperberg, S. S.; Danishevski, S.; Felner, A.; and Sturtevant, N. R. 2021. Iterative-deepening Bidirectional Heuristic Search with Restricted Memory. In *ICAPS*, 331–339.

Shperberg, S. S.; Felner, A.; Shimony, S.; Sturtevant, N.; and Hayoun, A. 2019a. Improving bidirectional heuristic search by bounds propagation. In *SoCS*, 106–114.

Shperberg, S. S.; Felner, A.; Sturtevant, N. R.; Shimony, S. E.; and Hayoun, A. 2019b. Enriching non-parametric bidirectional search algorithms. In *AAAI*, 2379–2386.

Siag, L.; Shperberg, S. S.; Felner, A.; and Sturtevant, N. 2023a. Front-to-end bidirectional heuristic search with consistent heuristics: enumerating and evaluating algorithms and bounds. In *IJCAI*, 5631–5638.

Siag, L.; Shperberg, S. S.; Felner, A.; and Sturtevant, N. R. 2023b. Comparing Front-to-Front and Front-to-End Heuristics in Bidirectional Search. In *SoCS*, volume 16, 158–162.

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

Thayer, J. T.; and Ruml, W. 2011. Bounded Suboptimal Search: A Direct Approach Using Inadmissible Estimates. In *IJCAI*, 674–679.