Position Paper: On the Impact of Direction-Selection in BAE*

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

Abstract

BAE*, and the independently developed DIBBS, are state-of-the-art bidirectional heuristic search algorithms that exploit heuristic consistency to efficiently prove solution optimality. Historically, BAE* has been studied with various direction-selection policies, determining whether to expand the next state from the forward or backward search. However, some of these policies expand nodes with an f-value exceeding the optimal solution cost, C^*, which clearly cannot be part of any optimal solution. In this position paper, we review direction-selection strategies in BAE* and bidirectional search more broadly, analyzing their impact on the behavior of the search. Additionally, we present a low-overhead solution that prevents the expansion of nodes with $f > C^*$ across all direction-selection strategies.

1 Intoduction

In optimal search, the task is to find the least-cost path between two vertices, $start$ and $goal$, in a given graph. *Unidirectional Heuristic Search* (UHS) algorithms, such as A* (Hart, Nilsson, and Raphael 1968), search for such a path by traversing the graph from $start$, constructing a search tree using a heuristic function as a guide. Specifically, A* prioritizes nodes in the search tree using the priority function $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of the least-cost path discovered so far from $start$ to n, and $h(n)$ is an estimation of the least-cost path from n to $goal$. Importantly, when the heuristic is *admissible*, i.e., never overestimating the cost from a vertex to $goal$, for every expanded node n, $f(n)$ is a lower bound on the least-cost path from $start$ to $goal$ that passes through n. Consequently, A* is guaranteed to return an optimal solution with cost C^* and to never expand nodes n for which $f(n) > C^*$.

Bidirectional Heuristic Search (BiHS) is an alternative to UHS that progresses simultaneously from $start$ (forward) and $goal$ (backward) until the two frontiers meet. Recent work on BiHS has presented theoretical studies and practical methods 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). BAE* (Sadhukhan 2013), and the identical DIBBS algorithm (Sewell and Jacobson 2021), are state-of-the-art BiHS algorithms that explicitly assume that consistent heuristics are provided. These assumptions allow for better pruning of nodes during the search and earlier termination, often leading to significantly fewer node expansions (and faster runtime) compared to A* and other BiHS algorithms (Alcázar, Riddle, and Barley 2020; Siag et al. 2023b).

One of the key decisions in BiHS, and for BAE* in particular, is determining which search direction (forward or backward) to choose when expanding the next node. Literature on BAE* explores various direction-selection policies (Alcázar, Riddle, and Barley 2020; Siag et al. 2023a), with the most popular being the *alternating* policy that alternates node-expansion in a round-robin fashion between forward and backward frontiers.

When visualizing the search for A* and BAE*, we discovered an interesting phenomenon: BAE* sometimes expands nodes n with $f(n) > C^*$. Such nodes are guaranteed not to be on the shortest path from $start$ to $goal$. While algorithms with an inconsistent heuristic sometimes need to expand nodes with $f(n) > C^*$ to avoid excessive additional expansions (Kaur and Sturtevant 2022), a consistent heuristic, as assumed in BAE*, ensures that such nodes are always redundant and should never be expanded. As it turns out, this behavior is caused by the direction-selection policy, which has unexpected effects that warrant further discussion. Our goal is to highlight these impacts.

In this paper, we review the direction-selection policies in BAE*, analyze their impact on the expansion of nodes with $f(n) > C^*$, and present a small-scale empirical evaluation comparing these policies. Notably, removing nodes with $f(n) > C^*$ during the search is not feasible, as C^* is not known *a priori*. Thus, we propose an efficient method to prevent the expansion of nodes with $f > C^*$ across all direction-selection strategies.

2 Definitions and Background

In BiHS, the objective is to find the least-cost path between $start$ and $goal$ in a given graph G. Let $c(x, y)$ denote the cost of a least-cost path between nodes x and y, and define $C^* \triangleq c(start, goal)$ as the optimal cost. BiHS performs a forward search (denoted as F) from $start$ and a backward search (denoted as B) from $goal$ until the two searches meet.

The algorithm typically maintains 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 (denoted as g_F, h_F, f_F for the forward search and g_B, h_B, f_B for the backward search). For a direction $D \in \{F, B\}$, the quantities f_D, g_D, and h_D represent the f-value, g-value, and h-value in that direction, respectively.

Most BiHS algorithms use 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 nodes $s \in G$. The heuristic h_F is *forward admissible* if and only if $h_F(s) \leq c(s, goal)$ for all $s \in G$. It is *forward consistent* if and only if $h_F(s) \leq c(s, s') + h_F(s')$ for all $s, s' \in G$. Similarly, backward *admissibility* and *consistency* are defined analogously.

2.1 BAE*

BAE* (Sadhukhan 2013; Alcázar, Riddle, and Barley 2020) and the equivalent DIBBS (Sewell and Jacobson 2021) are recent BiHS algorithms that assume heuristic consistency, leading to improved search performance.

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

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

where $h_{\overline{D}}$ is the heuristic in the direction opposite to D. The function $b_D(n)$ 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 the node with the minimal b_D-value. Additionally, BAE* terminates once the same state n is found on both open lists and the cost of the path from $start$ to $goal$ through n is less than or equal to LB_B, where LB_B is a lower bound on C^* given by:

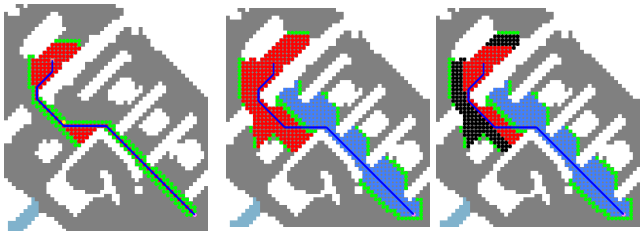
$$LB_B = \frac{bMin_F + bMin_B}{2} \tag{1}$$

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

Given a consistent heuristic, BAE* is proven to return an optimal solution. The function $b(n)$ is more informed than other priority functions, as it also considers $d(n)$. Consequently, BAE* has been shown to outperform common unidirectional and bidirectional algorithms across various domains (Alcázar, Riddle, and Barley 2020; Siag et al. 2023a), with improvements of up to an order of magnitude. Therefore, BAE* is considered a state-of-the-art BiHS algorithm for consistent heuristics.

In addition to the priority function, BiHS algorithms, including BAE* and DIBBS, need to employ direction-selection strategies to choose the direction D (either F or B) from which the node with the lowest b_D-value will be expanded. BAE* was originally presented with an *alternating* direction-selection policy, where D alternates between F and B after every node expansion (Sadhukhan 2013). Later, Alcázar, Riddle, and Barley (2020) applied another policy, known as Pohl's cardinality criteria, which was previously used in other BiHS algorithms (e.g., BS* (Kwa 1989)). Pohl's cardinality criterion chooses the direction D with the smallest open list, i.e., $D = \text{argmin}(|\text{OPEN}_\text{F}|, |\text{OPEN}_\text{B}|)$. Alcázar, Riddle, and Barley (2020) showed empirically that both direction-selection strategies have comparable performance, with slight variations across different domains.

DIBBS uses a different direction-selection strategy. It sets D to be the direction with the minimal b-value, i.e., $D = \text{argmin}(bMin_F, bMin_B)$. A tie-breaking policy in case both directions have a minimal b-value was not discussed by that paper. We refer to this direction-selection strategy as *min-b*. Interestingly, since DIBBS was developed independently from BAE*, their direction-selection strategies have not been compared analytically or empirically.



(a) A* (b) BAE* (c) BAE*: $f > C^*$

Figure 1: Visualization of nodes expanded by A* and BAE*.

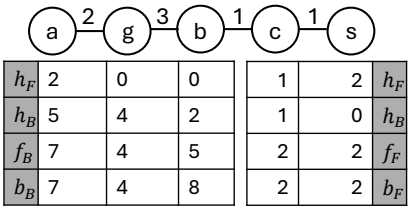
3 Visualization and Analysis of Direction-selection Policies in BAE*

The alternating direction-selection policy is widely used in BAE* due to its simplicity, strong performance, and its ability to prevent *starvation*, where nodes are expanded mostly from one direction. However, we identified a limitation of this policy while observing a search demonstration on the MoovingAI website.[1] The tool allows users to choose a search algorithm and problem configuration, and to visualize the nodes explored throughout the search process.

By comparing the node expansions of A* with those of BAE*, it became evident that some nodes expanded in the forward search of BAE* were never expanded by A*. An example of this can be seen in the snapshot from the demo shown in Figure 1. The figure illustrates the nodes expanded by A* (1a) and by BAE* (1b); nodes expanded in the forward search are shown in red, while those expanded by the backward search are in blue (applicable only to BAE*).

Upon reviewing these figures, it is clear that certain locations were expanded by BAE*'s forward search but not by A*. Figure 1c highlights these cases, showing the states expanded by BAE* with $f > C^*$ in black. This visualization of unnecessary expansions was added later to aid reader understanding, though the behavior is already evident from the first two figures. Since A* avoids expanding these locations (because their f-value exceeds C^*), they should not be expanded by any algorithm, as no optimal solution can pass through them. The fact that BAE* expands these nodes

[1] https://www.movingai.com/SAS/NBS/



h_F	2	0	0	1	2	h_F
h_B	5	4	2	1	0	h_B
f_B	7	4	5	2	2	f_F
b_B	7	4	8	2	2	b_F

Figure 2: BAE* expanding nodes with $f > C^*$ using the alternating direction-selection policy

suggests a potential issue with its search process, which we would not have discovered without the visualization.

Based on insights gained from the visualization, we have constructed a small example demonstrating that BAE* can expand nodes with an f-value greater than C^*. In this example, presented in Figure 2, the aim is to find the shortest path from s to g (of cost 5). Edge costs are shown above edges; heuristic values and priorities appear in a table below each node. Using the alternating direction-selection policy, BAE* first expands s ($start$) and g ($goal$), generating nodes a, b, and c. Assume, W.L.O.G., that in the alternating policy, nodes are expanded from the backward direction before the forward direction. At this stage, $\text{OPEN}_\text{B} = \{a, b\}$. Since $b_B(a) = 7 < b_B(b) = 8$, node a is selected for expansion. However, we observe that $f(a) > C^*$, indicating that a should never be expanded. A similar example can be constructed for Pohl's cardinality criterion by adding nodes with large b values connected from s.

By contrast, when using the min-b side-selection policy, node a will not be expanded. In contrast, when the min-b side-selection policy is used, node a will not be expanded. More generally, under the min-b side-selection policy in BAE*, any node n with $f_D(n) > C^*$ will never be expanded. Intuitively, since

$$\frac{bMin_F + bMin_B}{2} \geq \min(bMin_F, bMin_B)$$

and

$$\frac{bMin_F + bMin_B}{2} \leq C^*,$$

it follows that $\min(bMin_F, bMin_B) \leq C^*$.

Let $n = \text{argmin}(bMin_F, bMin_B)$. Since for every node n', it holds that $b(n') \geq f(n)$, we conclude that

$$f(n) \leq b(n) \leq C^*.$$

A broader observation is that if every node expanded using some priority function Pr_D in direction D satisfies: (i) $Pr_D(n) \geq f_D(n)$ and (ii) $Pr_D(n) \leq C^*$, then any node n with $f_D(n) > C^*$ will never be expanded.

These conditions are satisfied by most optimal BiHS algorithms, including BHPA (Pohl 1970), MM (Holte et al. 2017), NBS (Chen et al. 2017), DVCBS (Shperberg et al. 2019b), and GBFHS (Barley et al. 2018), all of which avoid expanding nodes with $f_D(n) > C^*$. However, condition (ii) may not hold for BAE* when using direction-selection policies other than min-b, potentially leading to the expansion of such nodes. This issue can also arise in other algorithms that adopt similar policies; for example, MM with the alternating policy may expand nodes with $f_D(n) > C^*$.

This analysis highlights the unintended inefficiencies that can arise from commonly used direction-selection policies in bidirectional search. We believe that raising awareness of these issues is crucial for the further development of efficient and reliable BiHS algorithms.

Policy	STP	Grid	Pancake		
			GAP	GAP-1	GAP-2
Alt.	2,707,405 (246)	15,888 (543)	8,992	**20,815**	**126,494**
Pohl	2,837,082 (958)	**14,144** (1,405)	**8,086**	35,130	147,365
min-b	**2,340,866**	17,017	9,099	23,772	141,454

Table 1: BAE* average nodes expansions

4 Empirical Comparision

The code for the empirical evaluation is publicly available (SPL-BGU 2024). Since all three aforementioned direction-selection strategies have never been empirically compared, we aim to evaluate their performance. To this end, we conducted the first empirical study across three domains:

- **16-Pancake Puzzle:** 100 hard instances (Valenzano and Yang 2017) using the GAP heuristic (Helmert 2010). To explore a range of heuristic strengths, we also evaluated GAP-n heuristics ($n \in \{1, 2\}$), where the n smallest pancakes are excluded from heuristic computation.
- **15-sliding tile puzzle (STP):** Korf's 100 instances using the Manhattan Distance (MD) heuristic.
- **Grid Pathfinding:** 156 maps from Dragon Age: Origins (DAO) (Sturtevant 2012), across different start and goal locations (totaling 15,980 instances), with the octile distance heuristic and diagonal moves costing 1.5.

Our experiment was conducted on a computing cluster equipped with AMD EPYC 7763 64-Core processors and 32GB of RAM per machine.

Notably, when a direction-selection strategy results in a tie between both directions—such as when both have the same minimal b-value—we break the tie by prioritizing the direction that was not selected in the previous expansion.

The results, presented in Table 1, show the average number of node expansions for each direction-selection policy. The average number of nodes expanded with $f > C^*$ is indicated in parentheses, except when it is zero, in which case the parentheses are omitted. Notably, this analysis can only be performed *a posteriori*, after determining C^*.

The results indicate that all strategies demonstrate comparable performance, with a different strategy being the most effective in each domain. Notably, the number of nodes expanded with $f_D(n) > C^*$ remains relatively low on average, reaching a maximum of 10% for Pohl's cardinality on grids. In certain individual instances, however, up to 84% of the nodes (4595 out of 5476) were expanded with $f_D(n) > C^*$.

5 Delayed Computation of Bounds

Using min-b as a direction-selection strategy is a valid approach that ensures nodes with $f_D(n) > C^*$ are never

expanded. However, in some domains, min-b is less efficient—for instance, it expands 20% more nodes in Grid compared to Pohl's cardinality policy. More broadly, supporting diverse direction-selection policies is a key aspect of BiHS. Therefore, our goal is to enable algorithms to use any direction-selection policy while preventing the expansion of nodes with $f_D(n) > C^*$, all without incurring significant computational overhead.

Previous work in bidirectional search has addressed the challenge of strengthening the lower bound during search to prune additional nodes (Shperberg et al. 2019a; Alcázar, Riddle, and Barley 2020; Siag et al. 2023a). A particularly relevant approach was presented by (Alcázar, Riddle, and Barley 2020), in which they track the current lower bound on the solution cost, denoted here as LB. They classify delayed nodes as those from which no solution of cost LB can be reached. This includes, for example, nodes with $f > LB$, among others. Since $LB \leq C^*$, this ensures that no nodes with $f > C^*$ will be expanded.

This approach can be applied to BAE* with an alternating policy by restricting expansion to nodes that could contribute to a solution matching the current LB. However, the method proposed by (Alcázar, Riddle, and Barley 2020), referred to as DBBS (not to be confused with DIBBS), requires maintaining nodes in buckets based on their f, h_F, and h_B values, and performing a linear scan of all buckets whenever LB increases to identify the relevant nodes. This process is computationally expensive and results in longer runtimes, despite the reduction in node expansions due to additional pruning. This inefficiency is particularly pronounced in domains with a large number of buckets, such as grid navigation.

An alternative approach we propose is to use a bi-level data structure (denoted as BL) inspired by the one used in NBS (Chen et al. 2017). Similar to (Alcázar, Riddle, and Barley 2020), a lower bound LB on the search is initialized as $\max(h_F(start), h_B(goal))$ and is continuously updated. The data structure consists of two priority queues for each search direction, where each priority queue maintains a disjoint set of nodes.

The first priority queue, $waiting_F$, is ordered by f_F in ascending order. All nodes with $f_B \leq LB$ are removed from $waiting_F$ and inserted into the second priority queue, $ready_F$, which is sorted by ascending b_F-values (as opposed to the NBS data structure, where the second priority queue is sorted by g_F-value). A similar pair of queues is maintained for the backward search. Nodes are expanded from $ready_F$ or $ready_B$, depending on the direction-selection policy.

Whenever the b-bounded lower bound, LB_b (Equation 1), computed using nodes in the ready queue, exceeds LB, or when one of the ready queues is depleted, LB is updated to the smallest f-value in the waiting queues:

$$LB = \min\left(\min_{n \in \text{waiting}_F}(f_F(n)), \min_{n \in \text{waiting}_B}(f_B(n))\right).$$

The search terminates once a solution of cost less than or equal to LB is found.

Since this approach avoids a linear scan of nodes and ensures that each node is inserted into each queue at most once, it efficiently filters out nodes with f-values exceeding C^*.

Policy	Expanded	Time	Nodes/Sec
BAE* (Pohl)	14,144 (543)	**0.0075**	1,807,497
BL-BAE* (Pohl)	**13,546**	**0.0075**	1,710,867
DBBS (Pohl)	13,740	1.6747	193,769
BAE* (Alt.)	15,888 (1,405)	0.0098	1,670,679
BL-BAE* (Alt.)	15,013	0.**0095**	1,579,366
DBBS (Alt.)	**14,582**	0.8190	221,172

Table 2: Results of delayed expansions

We empirically evaluated the results of the bi-level approach on the Grid domain, compared DBBS in terms of runtime and node expansions. The results are presented in Table 2. First, we observe that delaying nodes with $f_D(n) > LB$ results in no nodes being expanded with $f_D(n) > C^*$. Additionally, when using the BL data structure, there is a small reduction in overall node expansions compared to BAE* for both Pohl's cardinality and the alternating direction-selection policies. Furthermore, when examining the number of nodes expanded per second, we observe that BL introduces a moderate overhead of approximately 5%. In comparison to DBBS, we find that DBBS results in slightly fewer node expansions with the alternating policy, due to its use of additional bounds to delay more nodes; however, this comes at the cost of significantly higher runtime, making it impractical.

6 Conclusion and Discussion

In this work, we analyzed the impact of direction-selection policies in BAE*. We presented an insightful example that highlights the power of visualization as a tool for understanding search behavior. By visually analyzing the search process, we identified inefficiencies that were not apparent in existing empirical evaluations and theoretical analysis. Specifically, we showed that certain policies—particularly the commonly used alternating policy—can lead to the expansion of nodes with $f > C^*$, which cannot contribute to an optimal solution. This issue, absent in A*, underscores that direction selection is not just an implementation detail but a fundamental design choice that directly affects search efficiency.

A key goal of this paper is to raise awareness of these effects and promote a more critical examination of direction-selection policies. To address this, we proposed a bi-level priority queue mechanism that ensures nodes with $f > C^*$ are never expanded, effectively eliminating unnecessary expansions with minimal overhead while maintaining flexibility in direction selection.

More broadly, our results emphasize the importance of visualization and structured analysis for diagnosing and improving heuristic search algorithms. We aim to spark further research into the role of direction-selection policies in bidirectional search to refine algorithmic decisions and enhance search efficiency.

Acknowledgments

This work was supported by the Israel Science Foundation (ISF) grant #909/23 awarded to Shahaf Shperberg and Ariel Felner, 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. This work was also supported by the National Science and Engineering Research Council of Canada Discovery Grant Program and the Canada CIFAR AI Chairs Program.

References

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.

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

Barley, M.; Riddle, P.; López, C. L.; Dobson, S.; and Pohl, I. 2018. GBFHS: A Generalized Breadth-First Heuristic Search Algorithm. In *Eleventh Annual Symposium on Combinatorial Search*, 28–36.

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.

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.

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.

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

Kaur, J.; and Sturtevant, N. R. 2022. Efficient Budgeted Graph Search. *International Joint Conference on Artificial Intelligence (IJCAI)*.

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

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4): 193–204.

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.

SPL-BGU. 2024. BiHS Direction Choosing Code Repository. https://github.com/SPL-BGU/BiHS-Direction-Choosing.

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

Valenzano, R. A.; and Yang, D. S. 2017. An Analysis and Enhancement of the Gap Heuristic for the Pancake Puzzle. In *SOCS*, 109–117.