Front-to-Attractors: Modifying the Front-to-Front Heuristic in Bidirectional Search

Alvin Zou[1], **Muhammad Suhail Saleem**[1], **Maxim Likhachev**[1]

[1]Carnegie Mellon University
azou@andrew.cmu.edu, msaleem2@andrew.cmu.edu, maxim@cs.cmu.edu

Abstract

Heuristics play a central role in the performance of bidirectional search algorithms, which commonly rely on two main classes. *Front-to-end* (F2E) heuristics estimate the distance from a state s to the target of the search (the $goal$ for forward search or the $start$ for backward search). In contrast, *front-to-front* (F2F) heuristics estimate the distance from s to the opposite search frontier using a pairwise function $h(s, s')$, where s' ranges over frontier states. Although F2F heuristics are typically more informative and therefore reduce the number of node expansions, their reliance on extensive pairwise evaluations incurs substantial computational overhead. To address this limitation, we introduce a new heuristic class, *front-to-attractors* (F2A), that preserves much of the informativeness of F2F while dramatically reducing its computational cost. Rather than evaluating distances to all states on the opposite frontier, F2A estimates the distance from s to a small, dynamically maintained set of *attractors* in the opposite search direction. These attractors serve as a surrogate for the full frontier, enabling rich heuristic guidance at a fraction of the computational expense while maintaining the optimality guarantees offered by F2F. We evaluate F2A across multiple domains and show that it reduces the number of pairwise evaluations by up to $11.2\times$ compared to F2F, while achieving $4.8\times$ fewer node expansions than F2E on average.

Extended version — https://arxiv.org/abs/2606.07047

Introduction

Bidirectional heuristic search (Bi-HS) aims to find the least-cost path between two states by running a forward and backward search simultaneously, reducing effective search depth as the two frontiers meet. A central design choice in Bi-HS is the heuristic used to guide these searches. Most existing algorithms use *front-to-end* (F2E) heuristics, which estimate the cost from a state to the target of the current search direction (*goal* for forward search or *start* for backward search). Although inexpensive, F2E heuristics offer weak guidance because they treat the two searches independently and ignore the progress of the opposite frontier (i.e., Open list). In contrast, *front-to-front* (F2F) heuristics estimate the distance from a state to the opposite frontier using a pairwise function $h(s, s')$, where s' ranges over the states in the opposite frontier. F2F heuristics are more informative (Siag et al. 2023) and typically require fewer state expansions to find a solution. Their drawback is computational cost: evaluating the heuristic of a state in one direction requires pairwise evaluations between the state and all states in the opposite frontier.

Due to this overhead leading to long planning times, most work on improving Bi-HS focuses on F2E methods, proposing enhanced expansion rules, stronger termination conditions, and effective pruning strategies (Sadhukhan 2013; Holte et al. 2017; Shaham et al. 2017; Eckerle et al. 2017; Chen et al. 2017; Sewell and Jacobson 2021; Shperberg et al. 2019; Wang et al. 2025). Only a few approaches attempt to exploit or refine F2F heuristics (Felner et al. 2010; Lippi, Ernandes, and Felner 2012, 2016; Mayer and Krebsbach 2019; Alcázar 2021). Although F2F-based algorithms offer strong potential as their heuristics are more informative and often reduce the number of state expansions, they remain significantly slower in practice because heuristic evaluation scales quadratically with frontier size (Siag et al. 2023). Techniques that try to represent the frontier with a smaller set exist, but give suboptimal solutions (Lavasani et al. 2025).

To overcome the overhead associated with F2F heuristics, this manuscript introduces a new heuristic class, front-to-attractors (F2A), that retains much of the informativeness of F2F while greatly reducing its computational cost. The approach leverages the attractor representation introduced in Zou, Saleem, and Likhachev (2025). Instead of comparing s against all states on the opposite frontier, F2A evaluates distances to a small, dynamically maintained set of *attractors*. These attractors serve as a compact surrogate for the full frontier, enabling rich heuristic guidance at a fraction of the computational expense while preserving the optimality guarantees of F2F. We evaluate F2A across multiple domains, including 2D grid pathfinding, Sliding Tiles, and Pancake puzzle, and show that it reduces the number of pairwise heuristic evaluations by up to $11.2\times$ compared to F2F, while achieving $4.8\times$ fewer node expansions than F2E on average.

Background

A Bi-HS problem instance $I = (G, start, goal, h_F, h_B)$ consists of a graph $G = (V, E)$, a $start$ state, a $goal$ state, and a forward (F) and backward (B) heuristic h_F, h_B. The cost of the shortest path between state s and s' in G is denoted by $c(s, s')$. The objective is to find a path from $start$ to $goal$ with optimal cost C^*. Bi-HS algorithms perform two searches simultaneously: a forward search from $start$ and a backward search from $goal$. Each search direction

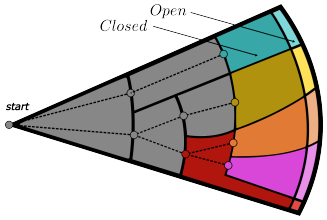

Figure 1: Attractors are shown as circles, with associated states in matching colors. The Open list is shown in lighter shades, and the remaining states are in the Closed list. Attractors with no associated frontier states are shown in gray.

$D \in \{F, B\}$ maintains its own Open list, $Open_D$. For a state s, $g_D(s), h_D(s), f_D(s)$ denote its g-, h-, and f-values in direction D. The opposite direction is denoted by OD.

front-to-end (F2E) heuristics use a forward heuristic $h_F : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ that estimates the cost from any state to $goal$ and a backward heuristic $h_B : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ that estimates the cost from any state to $start$. The f-value is computed as $f_D(s) = g_D(s) + h_D(s)$. *front-to-front* (F2F) heuristics (Sint and de Champeaux 1977) use a pairwise heuristic $h : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ that estimates the cost between any pair of states. The F2F heuristic of a state s in direction D is computed by estimating the cost between s and the opposite frontier. The f-value is $f_D(s) = g_D(s) + \min_{s' \in Open_{OD}}(h(s, s') + g_{OD}(s'))$. We use the same definitions for bi-admissibility and bi-consistency as (Eckerle et al. 2017).

Attractors Attractors (Zou, Saleem, and Likhachev 2025) were introduced as a memory-efficient alternative to storing the full Closed list in best-first search. They represent the Closed list sparsely by storing only an intelligently selected small set of states and using them as anchors for solution reconstruction. During search, every state added to the Open list is assigned to an attractor, and each attractor (except $start$) is linked to a parent attractor, forming a tree that serves as a compact surrogate for the full Closed list. Fig. 1 illustrates how attractors decompose the search space.

Given a distance function $h_{\text{dist}} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$, attractors are chosen so that the path to any state s in the Open list can be reconstructed by *greedy tracing*. Formally, let $a(s)$ denote the attractor assigned to s. Greedy tracing proceeds by repeatedly applying the update $s \leftarrow \arg\min_{s' \in Pred(s)} h_{\text{dist}}(s', a(s))$, until reaching the attractor $a(s)$, after which tracing continues recursively through parent attractors until $start$ is reached. Each such greedy segment corresponds to a portion of the current best path to s, which allows all non-attractor states to be safely discarded while still enabling full solution reconstruction. More details on how attractors are maintained are in the appendix.

Method

In F2F heuristics, computing the heuristic value of a state in one search direction requires comparing it against all states in the opposite frontier to obtain a valid underestimate of the optimal path cost. While this provides highly informative guidance, it also incurs substantial computational overhead. To reduce this cost, we introduce *front-to-attractors* (F2A) heuristics. Although attractors were originally proposed for sparsifying the Closed list, they also have an important structural property: they form a sparse set of ancestors for the states on the frontier. F2A leverages this structure by replacing comparisons against the entire opposite frontier with comparisons against a much smaller, dynamically maintained set of active attractors, denoted $Attrs_D$.

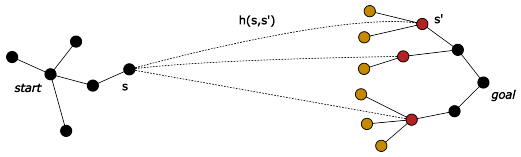

Figure 2: F2F heuristics estimate the distance to the opposite frontier (shown in yellow). F2A heuristics estimate the distance to the opposite attractors (shown in red).

As illustrated in Fig. 2, computing the F2F heuristic for a state s requires pairwise evaluations between s and all states in the opposite frontier (highlighted in yellow). In contrast, the F2A heuristic evaluates s only against the active attractors in the opposite direction that have at least one associated frontier state (highlighted in red). Formally, the F2A heuristic for a state s in direction D is computed as: $h_D(s) = \min_{s' \in Attrs_{OD}} \big(h(s, s') + g_{OD}(s')\big)$. Since all least-cost paths discovered so far must pass through their corresponding attractors, F2A preserves optimality guarantees while reducing the number of pairwise evaluations. [1]

Algorithm Overview

Bi-HS with F2A heuristics follows the standard Bi-HS loop used for F2F heuristics, requiring only minor additional bookkeeping to maintain the set of active attractors. The pairwise heuristic $h(s, s')$ must be bi-consistent. The overall procedure consists of the following steps, as outlined in Alg. 1. At each iteration, the search:

1. **Selects a search direction:** A common strategy ,which we use, is to select the direction D with the smaller Open list $Open_D$ (line 6), as noted by Holte et al. (2017).
2. **Expands a state:** The state s with the smallest f_D-value (higher g_D-value for tiebreaking) in $Open_D$ is selected for expansion (lines 7–8). Its successors are generated, assigned f-values, inserted into $Open_D$, and used to update U (lines 13–23). Each successor's heuristic is computed once upon generation by comparing the state to attractors in the opposite direction $Attrs_{OD}$ (line 20).
3. **Updates attractors:** When expanding s, if a better path to a successor s' is found, the attractor of s' is updated (line 16). The algorithm attempts to assign s' the same attractor as s to minimize the number of attractors maintained. This is allowed only if greedy tracing from s' toward this attractor leads through s (line 25). If this condition holds, s' inherits s's attractor (lines 27–28). Otherwise, s becomes the attractor for s' and is inserted into $Attrs_D$ (line 30). This update logic follows the same principles Zou, Saleem, and Likhachev (2025)[2]; details are omitted from the pseudocode for clarity.

[1] F2A keeps only the active attractors with associated states in the Open list (colored in Fig 1), rather than the full attractor tree.

[2] We also use the same tiebreaking strategy when paths of the same cost are found.

Algorithm 1: BI-HS WITH F2A HEURISTIC

1: **procedure** F2A BI-HS($start, goal$)
2: $Attrs_F \leftarrow \{start\}, Attrs_B \leftarrow \{goal\}$
3: $Open_F \leftarrow \{start\}, Open_B \leftarrow \{goal\}$
4: $U, L \leftarrow \infty$
5: **while** NOTTERMINATED(U, L) **do**
6: $D \leftarrow$ CHOOSEDIRECTION()
7: POP s with smallest f_D-value from $Open_D$
8: EXPAND(s, D)
9: $L \leftarrow$ UPDATELB()
10: REMOVE all $a \in Attrs_D$ that are unassigned
11: **procedure** EXPAND(s, D)
12: Move s from $Open_D$ to $Closed_D$
13: **for** $s' \in Succ_D(s)$ **do**
14: **if** $g_D(s') \geq g_D(s) + c(s, s')$ **then**
15: $g_D(s') = g_D(s) + c(s, s')$
16: UPDATEATTRACTOR(s, s', D)
17: **if** $s' \in Open_D \cup Closed_D$ **then**
18: remove s' from $Open_D \cup Closed_D$
19: **if** $s' \notin Open_D$ **then**
20: $h_D(s') = \min_{s'' \in Attrs_{OD}}(h(s', s'') + g_{OD}(s''))$ ▷ If not yet computed
21: INSERT s' in $Open_D$ with $g_D(s') + h_D(s')$
22: **if** $s' \in Open_{OD}$ **then**
23: $U \leftarrow \min(U, g_F(s') + g_B(s'))$
24: **procedure** UPDATEATTRACTOR(s, s', D)
25: $s''_{min} \leftarrow \text{argmin}_{s'' \in Pred(s')} \mathbf{h_{dist}}(s'', s.attractor)$
26: $s'.attractor \leftarrow s$
27: **if** $s''_{min} = s$ **then**
28: $s'.attractor \leftarrow s.attractor$
29: **else if** s not in $Attrs_D$ **then**
30: INSERT s in $Attrs_D$

By assigning attractors in this manner, $Attrs_D$ forms a sparse set of ancestor states through which all current best paths to the frontier states pass. This sparsity makes F2A heuristics much cheaper to compute while still providing strong guidance.

4. **Checks for termination:** The search terminates when the current best solution cost U is less than or equal to the lower bound L (optimal path has been found) or if the Open list becomes empty (no solution exists) (line 5).

Theoretical Properties

Our proof follows the same structure as Holte et al. (2017). Here we provide a high level sketch. A more detailed proof can be found in the appendix.

Definition 1. *For any optimal path $P = s_0, s_1, ...s_n$ from $start$ (s_0) to $goal$ (s_n), let i be the largest index such that $s_k \in Closed_F \; \forall k \in [0, i-1]$, and let j be the smallest index such that $s_k \in Closed_B \; \forall k \in [j+1, n]$. We say P "has not been found" if $i < j$ and P "has been found" otherwise.*

Theorem 1. *If, at the beginning of an iteration of Alg. 1, there exists an optimal path P from $start$ to $goal$ that has not been found, then $L \leq C^*$.*

Proof Sketch. If there exists an optimal path that has not been found, there will be a node in P, s_i, in $Open_F$ with $g_F(s_i) = c(start, s_i)$ and a node in P, s_j, in $Open_B$ with $g_B(s_j) = c(s_j, goal)$. Then $C^* = g_F(s_i) + c(s_i, s_j) + g_B(s_j) \geq g_F(s_i) + h(s_i, s_j) + g_B(s_j)$ (admissibility). With the F2A heuristic, $f_F(s) = g_F(s) + \min_{a \in Attr_B}\big(h(s, a) + g_B(a)\big)$. By construction of attractors, for every $s' \in OPEN_B$ there exists $a \in Attr_B$ with $g_B(s') = c(s', a) + g_B(a)$. Since $h(s, a) \leq h(s, s') + c(s', a)$ (consistency), $h(s, a) + g_B(a) \leq h(s, s') + g_B(s')$. Taking minima over attractors yields $f_F(s) \leq g_F(s) + \min_{s' \in OPEN_B}\big(h(s, s') + g_B(s')\big)$. Therefore, $L = \min_{s \in OPEN_F} f_F(s) \leq \min_{s \in OPEN_F, s' \in OPEN_B}\big(g_F(s) + h(s, s') + g_B(s')\big) \leq g_F(s_i) + h(s_i, s_j) + g_B(s_j) \leq C^*$. □

Theorem 2. *If there exists a path from $start$ to $goal$, Alg. 1 will return $U = C^*$ (an optimal path) upon termination.*

Proof Sketch. Theorem 1 implies that $L \leq C^*$ until all optimal paths have been found. And since $U > C^*$ until the first optimal path from $start$ to $goal$ is found (due to the process by which U is updated), at which point $U = C^*$, this means that if a path does exist, when Alg. 1 terminates $U = C^*$. □

This implies that when Alg. 1 terminates at least one optimal path has been found, indicating that Bi-HS with F2A heuristics is both complete and optimal.

Practical Considerations

Degenerating to F2E If $goal$ is included in the backward attractor set $Attrs_B$, then for any state s in the forward direction we have $h(s, goal) \leq h(s, s') + g_B(s') \; \forall \, s' \in Attrs_B$, provided that h is consistent. This implies that the F2A heuristic degenerates to F2E when the goal (or start) is an attractor, reducing the informativeness of the heuristic.

Optimizations to preserve heuristic informativeness. In general, attractors become less informative as they drift farther from the frontier. To mitigate this, we introduce two optimizations. Both rely on a threshold δ applied to the difference in g-values between an attractor and its assigned states.

1. **New Attractor (NA).** When a successor s' inherits the attractor of its parent s, we check whether the difference between $g(s')$ and $g(s.\text{attractor})$ exceeds δ. If so, s is assigned as the new attractor for s'. This ensures the active attractor set is a good approximation of the search frontier by keeping attractors close.
2. **Associated States (AS).** For each attractor, we maintain the subset of its assigned frontier states whose g-values differ from the attractor's by more than δ. If this set is nonempty, we use the associated states in place of the attractor for heuristic computation. In effect, when an attractor becomes too far from the frontier to serve as an informative surrogate, we fall back to using its frontier states for heuristic evaluation.

Experiments

We evaluated F2A on three standard benchmark domains commonly used in Bi-HS studies. **(1) 2D Grid Pathfinding:** Evaluated on the DAO "brc" and maze maps with five corridor widths (Sturtevant 2012), using 4-connected unit-cost

Planner	DAO			Maze			Pancake Gap-1			Sliding Tiles		
	RT (ms)	**Exp** (10^3)	**HE** (10^3)	**RT** (ms)	**Exp** (10^3)	**HE** (10^3)	**RT** (ms)	**Exp** (10^3)	**HE** (10^3)	**RT** (ms)	**Exp** (10^3)	**HE** (10^3)
A*	47	17	17	534	185	185	220	6.5	67	2745	340	605
BAE* F2E	35	20	21	258	143	143	32	0.695	8	1205	151	296
VBi-HS F2E	85	25	25	1100	280	281	175	4.8	53	2746	307	577
VBi-HS F2F	91	14	2865	1043	134	34,907	26,545	0.541	26,189	118,263	20	92,775
VBi-HS F2A (NA)	76	16	525	778	136	4897	778	4.8	53	19,161	108	75,500
VBi-HS F2A (AS)	119	17	1273	1148	139	15,275	14,846	0.583	14,782	76,174	20	232,716
VBi-HS F2A (no opt)	76	18	278	660	138	3120	756	4.8	53	4374	307	593
NBS F2E	73	21	21	558	147	147	168	4.7	54	3111	290	548
NBS F2F	99	16	1794	934	135	15,709	35,068	0.828	35,759	344,803	37	332,158
NBS F2A (NA)	85	18	319	819	138	3545	517	4.7	183	31,832	94	143,202
NBS F2A (AS)	553	21	12,764	16,553	146	277,741	50,756	1.1	51,659	1,301,147	60	780,339
NBS F2A (no opt)	84	18	319	745	138	3545	449	4.7	104	5964	290	4306

Table 1: Results for planners across domains. F2A employ optimizations ("no opt" means no optimization) that vary by domain.

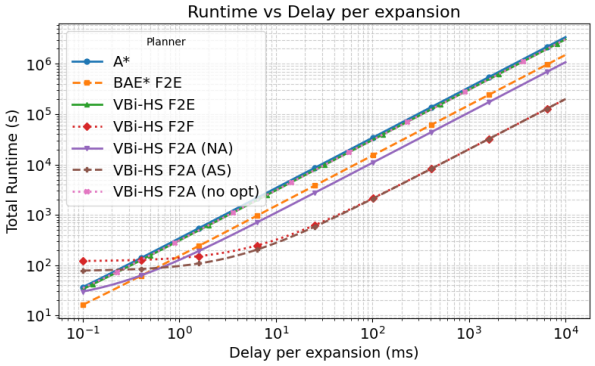

Figure 3: Sliding Tiles with an added delay per expansion.

actions and the Manhattan Distance heuristic. **(2) 15 Sliding Tiles:** 20 instances (fewest nodes) from (Korf 1985), using the Manhattan Distance heuristic. **(3) 14-Pancake Puzzle:** 50 random instances using the GAP-1 heuristic (GAP heuristic (Helmert 2010) but ignoring the smallest pancake.)

We compared F2A (with and without the NA and AS optimizations) to F2E and F2F using both vanilla Bidirectional Heuristic Search (VBi-HS) and Near-optimal Bidirectional Search (NBS) (Chen et al. 2017). VBi-HS denotes the framework used to instantiate F2E, F2F, and F2A (i.e., Alg. 1 with different heuristic formulations), following the ideas of (Pohl 1969) and (Sint and de Champeaux 1977). Since NBS was originally designed for F2E heuristics, we extended it to support F2F and F2A by changing the heuristic used to compute f-values while keeping the same pairwise lower bound. BAE* (Sadhukhan 2013) is a state-of-the-art Bi-HS algorithm but assumes fixed heuristic values assigned at node generation, making it unclear how to extend it to dynamic heuristics like F2F and F2A that depend on the evolving opposite frontier. However, we include BAE* with F2E heuristics and A* (Hart, Nilsson, and Raphael 1968) as references. The same search direction policy is used for the Bi-HS algorithms. We report results using $\delta = 20$ for both NA and AS on the grid maps, and $\delta = 4$ for Sliding Tiles and Pancake. For each planner we report mean runtime (RT), number of node expansions (Exp), and number of heuristic evaluations (HE), presented in Table 1. Our evaluation focuses on how different heuristic classes trade off computation cost against search guidance, rather than on identifying a dominant heuristic. Additional discussion on the experimental results is included in the appendix.

Grid Maps With VBi-HS, both F2A (no opt) and F2A (NA) achieve the fastest runtimes among all variants. The improvement is primarily driven by a substantial reduction in heuristic evaluations—10.3× fewer on DAO and 11.2× fewer on Maze maps compared to F2F—while still outperforming F2E in terms of expansions (1.4× and 2.0× fewer). Under NBS, the F2A variants also require fewer heuristic evaluations than F2F and produce fewer expansions than F2E. As expected, F2A reduces heuristic evaluations versus F2F while preserving informativeness, yielding fewer expansions than F2E; this trend holds across domains.

Sliding Tiles Naive F2A degenerates to F2E in this domain. However, incorporating the NA optimization makes F2A substantially more informative. With VBi-HS, expansions decrease by $2.84\times$ relative to F2E, and by $3.08\times$ under NBS. Heuristic evaluations decrease by $1.23\times$ and $2.32\times$ compared to F2F, for VBi-HS and NBS respectively. The AS optimization further reduces expansions but incurs a much higher number of heuristic evaluations.

To better understand when F2A heuristics provide benefits, we conducted an additional experiment in which an artificial delay was added to every state expansion. The resulting runtime curves, shown in Fig. 3, reveal a performance window after 0.6 ms where F2A (AS) achieves the best runtime across all planners. This study suggests the existence of domains in which the trade-off between heuristic computation cost and expansion cost aligns favorably for F2A, enabling it to outperform F2E, F2F, and A* in total runtime.

Pancake F2A also degenerates to F2E in the Pancake domain. The AS optimization substantially increases the informativeness of F2A. With VBi-HS, expansions decrease by $8.23\times$ relative to F2E, and by $4.21\times$ under NBS.

Conclusion

We introduced front-to-attractors (F2A), a heuristic for bidirectional search that preserves much of the guidance of front-to-front (F2F) while significantly reducing computational cost. By estimating the distance to a set of attractors rather than all frontier states, F2A requires minimal overhead and integrates easily into existing algorithms. Across multiple domains, it reduced heuristic evaluations by up to $11.2\times$ over F2F and achieved $4.8\times$ fewer expansions than F2E, while maintaining competitive runtimes, making it a practical approach for scalable bidirectional search.

Acknowledgments

The research was supported by the National Science Foundation by grant IIS-2328671. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

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 *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 7–15.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-end bidirectional heuristic search with near-optimal node expansions. *arXiv preprint arXiv:1703.03868*.

Eckerle, J.; Chen, J.; Sturtevant, N.; Zilles, S.; and Holte, R. 2017. Sufficient conditions for node expansion in bidirectional heuristic search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, 79–87.

Felner, A.; Moldenhauer, C.; Sturtevant, N.; and Schaeffer, J. 2010. Single-frontier bidirectional search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, 59–64.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Helmert, M. 2010. Landmark heuristics for the pancake problem. In *Proceedings of the International Symposium on Combinatorial Search*, volume 1, 109–110.

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. *Artificial Intelligence*, 252: 232–266.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1): 97–109.

Lavasani, S.; Siag, L.; Shperberg, S. S.; Felner, A.; and Sturtevant, N. R. 2025. Anchor Search: A Unified Framework for Suboptimal Bidirectional Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 27045–27053.

Lippi, M.; Ernandes, M.; and Felner, A. 2012. Efficient single frontier bidirectional search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 3, 105–111.

Lippi, M.; Ernandes, M.; and Felner, A. 2016. Optimally solving permutation sorting problems with efficient partial expansion bidirectional heuristic search. *AI Communications*, 29(4): 513–536.

Mayer, L. E.; and Krebsbach, K. D. 2019. Front-to-Front Bidirectional Best-First Search Reconsidered. In *FLAIRS*, 14–19.

Pohl, I. 1969. *Bi-directional and heuristic search in path problems*. Ph.D. thesis, Department of Computer Science, Stanford University.

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

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

Shaham, E.; Felner, A.; Chen, J.; and Sturtevant, N. 2017. The minimal set of states that must be expanded in a front-to-end bidirectional search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 8, 82–90.

Shperberg, S. S.; Felner, A.; Sturtevant, N. R.; Shimony, S. E.; and Hayoun, A. 2019. Enriching non-parametric bidirectional search algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2379–2386.

Siag, L.; Shperberg, S.; Felner, A.; and Sturtevant, N. R. 2023. Comparing Front-to-Front and Front-to-End Heuristics in Bidirectional Search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 16, 158–162.

Sint, L.; and de Champeaux, D. 1977. An Improved Bidirectional Heuristic Search Algorithm. *J. ACM*, 24(2): 177–191.

Sturtevant, N. 2012. Benchmarks for Grid-Based Pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2): 144 – 148.

Wang, Y.; Weiss, E.; Mu, B.; and Salzman, O. 2025. Bidirectional search while ensuring meet-in-the-middle via effective and efficient-to-compute termination conditions. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 8987–8995.

Zou, A.; Saleem, M. S.; and Likhachev, M. 2025. Attractor-based Closed List Search: Sparsifying the Closed List for Efficient Memory-Constrained Planning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 9031–9038.