Meeting at the Border of Two Separate Domains

Alexandru Paul Tabacaru, Dor Atzmon, Ariel Felner

Ben-Gurion University of the Negev
tabacaru@post.bgu.ac.il, dorat@post.bgu.ac.il, felner@bgu.ac.il

Abstract

To transmit information or transfer an object, two agents may need to reach the same location and meet. Often, such two agents operate in two separate environments and they can only meet at border locations. For example, a ship, sailing in the sea, needs to meet a truck traveling on land. These two agents are able to meet only at the shoreline. We call this problem the *Meeting at the Border* problem (MATB). In MATB, the optimal meeting location at the border is required, where the cost of a meeting location is the sum of the two shortest paths to that location. We show how to optimally solve MATB with heuristic search and suggest a novel heuristic function that estimates the cost of meeting at the border. Indeed, our new heuristic significantly enhances search algorithms in 2D and 3D domains.

Introduction and Related Work

In many cases, two agents must arrive at the same location in order to transmit information or transfer an object. However, the agents may not be able to meet at any location as the two agents operate in two separate environments. Outdoors, each of two such agents can be either a train on railway tracks, a truck on the road, a ship in the sea, or a drone in the sky. Indoors, the agents can be two different types of robots. For example, in a warehouse, one robot delivers items from shelves to a robot that travels on the ground. In all these cases, the agents can only meet in specific locations at the joint border of the two environments. We call the problem of finding a meeting location, in this border, the *Meeting at the Border* problem (MATB). The cost of a possible meeting location is the sum of the two shortest paths to that location. We are interested in finding the optimal (minimal-cost) meeting location, among all locations on the border.

In this paper, for optimally solving MATB, we consider the well-known unidirectional heuristic search algorithm *A** (Hart, Nilsson, and Raphael 1968), and the bidirectional heuristic search algorithm *MM* (Holte et al. 2017) (details on each algorithm, in the context of MATB, are provided below). Search algorithms usually use a *Front-To-End* heuristic function to estimate the remaining cost of reaching a goal location from a given location.[1]

[1]Bidirectional search algorithms can also use a *Front-To-Front* heuristic function that estimates the cost of meeting with a frontier of the other side of the search. We do not focus on such heuristics.

The main contribution of this paper is a novel heuristic function that enhances heuristic search algorithms for solving MATB. This heuristic function is a *Front-To-Border-to-End* heuristic, which estimates the cost of meeting at the border. We also propose a fast method for calculating this heuristic. We experimentally evaluate A* and MM with our new heuristic function in 2D and 3D domains. Our experiments show the significant benefit of our new function.

The problem of finding a meeting location for multiple agents was investigated in the past (Yan, Zhao, and Ng 2015; Atzmon et al. 2020, 2021). However, it is assumed that the agents can meet at any location. Izmirlioglu et al. (2017) considered a multi-agent meeting problem where the agents must pass through specific locations before the meeting. This is different from meeting in one of the locations on the border. Other works showed how to find meeting locations in continuous Euclidean spaces (Cooper 1968; Chen 1984; Rosing 1992; Lanthier, Nussbaum, and Wang 2005).

There is also some resemblance between MATB and *Perimeter Search* (Dillenburg and Nelson 1994), which is a bidirectional search that creates a fixed perimeter from one side of the search while the other side of the search progresses until it meets the perimeter. However, it results in a shortest path from the start to the goal while MATB aims to find two shortest paths to the border.

The Meeting at the Border Problem

The MATB problem receives as input two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, two start locations $s_1 \in V_1$ and $s_2 \in V_2$, and a set of border locations $B = V_1 \cap V_2$ that belong to both graphs. We denote the cost of traversing an edge $e \in E_1 \cup E_2$ by $C(e)$. Each agent must stay in its own graph and is not allowed to move to the other graph, except for border vertices which belong to both graphs. Let ϵ_1 and ϵ_2 denote the minimal edge cost in G_1 and G_2, respectively, and $\epsilon = \min(\epsilon_1, \epsilon_2)$ is the minimal edge cost among the two graphs. In G_1, two locations $v_x, v_y \in V_1$ are adjacent iff there is an edge $(v_x, v_y) \in E_1$ (similarly in G_2). A *solution* $\Pi = (\pi_1, \pi_2)$ to MATB consists of two paths $\pi_1 = (s_1, \ldots, v_b)$ and $\pi_2 = (s_2, \ldots, v_b)$ where any two consecutive locations in π_1 or π_2 are adja-

cent and $\pi_1 \cap \pi_2 = v_b \in B$. The cost $C(\pi)$ of a path $\pi = (v_1, \ldots, v_n)$ is the sum of costs of all edges traversed in π, i.e., $C(\pi) = \sum_{i=1}^{n-1} C((v_i, v_{i+1}))$. Therefore, the cost of solution $\Pi = (\pi_1, \pi_2)$ is $C(\Pi) = C(\pi_1) + C(\pi_2)$. An *optimal* solution Π^* has the minimal cost among all solutions.

Solving MATB with Heuristic Search

In this section, we elaborate on how MATB can be optimally solved with heuristic search. We start by defining MATB as a unidirectional search (Uni-HS). Then, we extend our definition to bidirectional search (Bi-HS). Finally, we explain how all searches can exploit heuristic functions in MATB.

Unidirectional Heuristic Search (Uni-HS)

In MATB, we receive two start locations. Thus, to perform Uni-HS, we start the search from one of the start locations and set the other as a goal location. We next describe the corresponding state-space.

Node. Each *Node* in the search space contains (1) a vertex location ($Node.v$), (2) a back pointer to the location $Node.v$ is reached from ($Node.prev$), and (3) a Boolean flag that marks if the border is already crossed ($Node.crossed$). Crossing the border by the search corresponds to performing a backward search to the other start location. Each node N is associated with a g-value and an h-value, where $g(N)$ is the cost of reaching location $N.v$ and $h(N)$ estimates the cost of reaching the goal location from location $N.v$. The Boolean flag, defined above, is mainly used for the heuristic function, as explained below.

Root. The search starts by initializing a *Root* node with (1) one of the start locations (e.g., $Root.v \leftarrow s_1$), (2) $Node.prev \leftarrow Null$, and (3) $Node.crossed \leftarrow False$. Experimentally, we saw that starting the search from the side with the higher minimal edge cost (ϵ_1 or ϵ_2) results in lower number of expansions and lower runtime. Therefore, in our experiments below, we always start from that side of the search. The higher cost side approaches the optimal cost faster and thus expands fewer nodes. *Root* is inserted into the open list (OPEN), which is ordered according to the prioritization of the selected search algorithm.

Goal. The *Goal* node is a node that contains the location of the other original start location that was not selected for the *Root* (e.g., $Goal.v = s_2$).

Expansion Cycle. Repeatedly, we select from OPEN the node N with the lowest f-value. In our experiments, we consider the A* algorithm, which uses $f(N) = g(N) + h(N)$. When node N is expanded, for each location adjacent to location $N.v$, a node N' is created, with $g(N') \leftarrow g(N) + C((N.v, N'.v))$. To prevent nodes from being re-expanded, we maintain a closed list (CLOSED). For each newly created node N', we check if $\exists N'' \in$ OPEN $\cup$ CLOSED s.t. $N'.v = N''.v$. In case it is false, N' is inserted into OPEN. Otherwise, if $g(N'') \leq g(N')$, N' is pruned and, if $g(N'') > g(N')$, N' is inserted into OPEN while N'' is extracted from OPEN or CLOSED. We track the current path by setting $N'.prev \leftarrow N$. If $N'.v \in B$, we set $N'.crossed \leftarrow True$. Otherwise, we set $N'.crossed \leftarrow N.crossed$. The cycle halts when the goal is expanded or when OPEN is empty.

Bidirectional Heuristic Search (Bi-HS)

In contrast to Uni-HS, Bi-HS algorithms progress from both directions of the search until frontiers of the two searches meet and the optimal solution can be determined (Holte et al. 2017). We maintain two open lists OPEN_F and OPEN_B for the forward and backward sides of the search, respectively. Thus, two search trees are created, one for each side. Similarly, we keep CLOSED_F and CLOSED_B.

Node. Nodes in Bi-HS are defined similarly to Uni-HS. However, here, $g_F(N)$ of node N in OPEN_F (resp. OPEN_B) denotes the cost of reaching location $N.v$ from the start location s_1 (resp. s_2) and $h_F(N)$ estimates the cost of getting to the other start location s_2 (resp. s_1) from $N.v$.

Roots. The two roots $Root_F$ and $Root_B$ are set similarly to the unidirectional case, but with the two start locations of MATB, i.e., $Root_F.v \leftarrow s_1$ and $Root_B.v \leftarrow s_2$. Each of the two open lists is initialized with its corresponding root node.

Prioritizing Nodes. Following MM (Holte et al. 2017), to guarantee that the search frontiers meet in the middle, nodes N in OPEN_F (resp. OPEN_B) are ordered by $pr_F(N) = \max(f_F(N), 2g_F(n) + \epsilon)$. Let $prmin_F$ and $prmin_B$ be the minimum priority on OPEN_F and OPEN_B. MM expands next a node with priority $C = \min(prmin_F, prmin_B)$.

Halting Condition. Instead of a goal node, here, we have a halting condition. When a location v is generated from both sides of the search, we know the cost of a solution in which one of the agents passes through v. We update an upper bound variable U (initialized with ∞) with the lowest cost found. Following MM (Holte et al. 2017), the search halts when $U \leq \max(C, fmin_F, fmin_B, gmin_F + gmin_B + \epsilon)$, where $fmin_F$ and $fmin_B$ are the lowest f values in OPEN_F and OPEN_B, and $gmin_F$ and $gmin_B$ are the lowest g values.

Importantly, when applying Bi-HS on MATB, the two search frontiers may cross the border and search backwards in the opposite graph (i.e., towards the other start location). Thus, the two search frontiers may meet in any location, not necessarily in a border location. However, the solution returned represents two paths that start at locations s_1 and s_2 and meet in a border location.

Heuristic Function (Front-to-End)

For a given node N, the heuristic function $h(N)$ estimates the cost from location $N.v$ to a goal location s ($= h(N.v, s)$). Let $h^*(N)$ be the exact cost of the optimal path from location $N.v$ to the goal (the other start location). A heuristic function h is *admissible* iff it never overestimates, i.e., $\forall N : h(N) \leq h^*(N)$. Given an admissible heuristic function, it is guaranteed for both A* and MM to return an optimal solution. While the edge costs are different between the two domains, in many cases it is possible to estimate the distance between the two locations $N.v$ and the goal location, e.g., by Manhattan distance, Octile distance, or Straight-line distance. Let $d(v_x, v_y)$ denote an admissible estimate on the number of edge traversals needed for reaching from v_x to v_y. Assume N is a node in OPEN_F (resp. OPEN_B) such that $Node.crossed = False$. Thus, $h(N) = \epsilon \cdot d(N.v, s_2)$ (resp. $h(N) = \epsilon \cdot d(N.v, s_1)$)

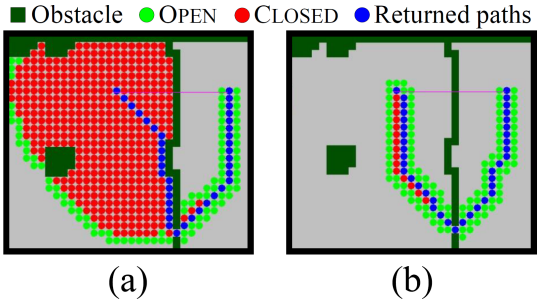

Figure 1: (a) FE heuristic and (b) FBE heuristic examples.

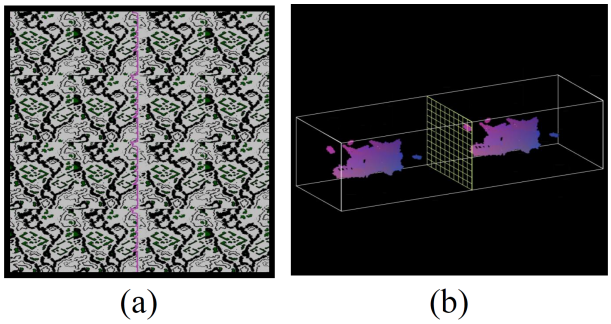

Figure 2: (a) 2D Cauldron map. (b) 3D Complex map.

is an admissible heuristic for N, where $\epsilon = \min(\epsilon_1, \epsilon_2)$. When $Node.crossed = True$, $h(N) = \epsilon_2 \cdot d(N.v, s_2)$ (resp. $h(N) = \epsilon_1 \cdot d(N.v, s_1)$) is also an admissible heuristic, as all edges left are in G_2 (resp. G_1). This is a *Front-to-End* (FE) heuristic as it estimates the cost to the end location.

Front-to-Border-to-End Heuristic

Assume two start locations in MATB that are close to each other while the closest border location is far from both start locations. In this case, a front-to-end heuristic function, as presented above, provides an extremely inaccurate estimate for MATB. Figure 1(a) presents an example of the case described above. The vertical green line separates two 8-connected grids. The blue dots denote the solution returned by A*, started from the left side. The light green and red dots denote the nodes that are in OPEN and CLOSED, respectively. There is only one border location at the bottom of the figure, where the blue dot of the solution passes through the vertical green line. Here, A* had to expand 450 nodes.

Hence, we develop the *Front-to-Border-to-End* (FBE) heuristic function. This heuristic estimates the cost of getting to one of the border locations and then to the goal. Given a node N in OPEN_F such that $N.crossed = False$, the agent may need to arrive at any location b in the border B. For a given border location $b \in B$, the following value is a lower bound of the optimal solution in which an agent is at location $N.v$ and the agents meet at border location b:

$$h(N, b) = h(N.v, b) + h(b, s_2). \quad (1)$$

Therefore, we suggest the following heuristic function:

$$h(N) = \min_{b \in B} h(N, b). \quad (2)$$

This heuristic estimates the cost for arriving at each location in the border. Then, it takes the minimal value among all border locations. Because the agent must get to one of the border locations, it is easy to see that the FBE heuristic is admissible. Figure 1(b) shows the same example for executing A* with FBE heuristic. Clearly, significantly fewer nodes are expanded (only 53 expansions).

In fact, the FBE heuristic is always more informed than FE. For a given node N that has not crossed the border ($N.crossed = False$), the two heuristic functions return the same heuristic value only if (1) the two domains have the same minimal edge cost, namely, $\epsilon_1 = \epsilon_2$, (2) the two domains have a similar distance estimation d, and (3) there is a border location between location $N.v$ and the goal location s_2, such that $d(N.v, s_2)$ is the actual number of edge traversals needed to reach the goal location.

Enhanced FBE (E-FBE)

While FBE is more informed than FE, it obviously requires more computation time as all border locations are considered. Thus, we develop the enhanced version E-FBE that calculates FBE without going over all border locations.

When the search starts, we create a static list BO, initialized with the following value for each border location b: $BO(b) \leftarrow h(s_1, b) + h(b, s_2)$. This value is a lower bound of an optimal solution in which the agents meet at location b. We order BO in ascending order by this value. Then, when the heuristic needs to be calculated for a given node N whose $N.crossed = False$, we iterate over the border locations in BO according to their order and calculate $h(N, b)$ (Equation 1). We set an upper bound UB (initialized with ∞) with $UB \leftarrow \min(UB, g(N)+h(N, b))$. When $UB \leq BO(b)$ for a border location b, the heuristic calculation halts and $UB - g(N)$ is returned as the heuristic value of node N (its f value is UB).

As was calculated at the beginning of the search, all other border locations that were not considered in the heuristic calculation have a higher cost for a meeting. Therefore, UB is set with the lowest value, without scanning these locations.

Experimental Study

To test our new heuristic function for MATB, we consider the 2D Cauldron map and the 3D Complex map, available in the `movingai` repository (Sturtevant 2012; Brewer and Sturtevant 2018), and random 3D maps. In the 2D map, we consider an 8-neighbor movement, where moving vertically or horizontally costs ϵ_1 and ϵ_2 (in G_1 and G_2) and moving diagonally costs $\sqrt{2}\epsilon_1$ and $\sqrt{2}\epsilon_2$. In 3D maps, we consider a 26-neighbor movement. Moving across one axis is equal to moving vertically or horizontally in 2D; moving across two axes is equal to moving diagonally; moving across three axes costs $\sqrt{3}\epsilon_1$ and $\sqrt{3}\epsilon_2$.

In our experiments, we explore the impact of two attributes on Uni-HS (A*) and Bi-HS (MM) with the different heuristics (FE, FBE, and E-FBE). The two attributes we examine are: (1) the *weight factor* (WF) between ϵ_1 and ϵ_2, i.e.,

	Border Density (BD)								Weight Factor (WF)							
	5%	10%	40%	100%	5%	10%	40%	100%	1	2	4	10	1	2	4	10
	#Expansions (millions)				Runtime (s)				#Expansions (millions)				Runtime (s)			
A*+FE	1.73	1.73	1.71	1.72	5.7	5.7	5.8	**5.8**	**1.12**	1.62	1.81	1.92	7.3	**6.6**	6.2	6.3
A*+FBE	**0.77**	**0.78**	**0.77**	**0.78**	5.5	5.9	7.6	11.2	**1.12**	**0.91**	**0.78**	**0.72**	10.0	7.9	6.5	5.5
A*+E-FBE	**0.77**	**0.78**	**0.77**	**0.78**	**5.0**	**5.1**	**5.3**	6.0	**1.12**	**0.91**	**0.78**	**0.72**	8.7	6.8	**5.8**	**5.3**
MM+FE	5.05	5.03	5.00	5.00	12.4	12.9	14.2	15.1	1.30	3.56	5.16	6.01	**4.4**	10.0	13.4	15.1
MM+FBE	1.57	1.57	1.56	1.57	7.7	9.3	19.1	39.2	1.30	1.42	1.72	2.70	9.4	10.9	13.8	22.2
MM+E-FBE	1.57	1.57	1.56	1.57	6.3	6.5	7.7	9.8	1.30	1.42	1.72	2.70	6.5	**6.6**	7.6	11.4

Table 1: Number of expansions (in millions) and average runtime (in seconds) on the 2D Cauldron map.

	Border Density (BD)								Weight Factor (WF)							
	5%	10%	40%	100%	5%	10%	40%	100%	1	2	4	10	1	2	4	10
	Success Rate				Runtime (s)				Success Rate				Runtime (s)			
A*+FE	49	49	**50**	49	92.7	93.1	91.5	91.8	49	48	**50**	**50**	13.0	88.2	77.7	87.4
A*+FBE	**50**	**50**	**50**	**50**	2.5	2.5	2.7	2.9	49	**50**	**50**	**50**	14.4	2.4	1.4	**1.2**
A*+E-FBE	**50**	**50**	**50**	**50**	2.7	2.7	2.9	3.0	49	**50**	**50**	**50**	14.6	2.1	1.5	1.3
MM+FE	9	9	7	6	284.5	285.3	287.9	288.0	**50**	16	7	1	**10.3**	255.5	289.3	300.0
MM+FBE	**50**	**50**	**50**	49	3.2	5.2	13.9	36.4	**50**	**50**	**50**	**50**	24.2	2.3	3.4	5.5
MM+E-FBE	**50**	**50**	**50**	**50**	**1.6**	**1.6**	**1.5**	**1.8**	**50**	**50**	**50**	**50**	11.4	**0.6**	**1.2**	2.0

Table 2: Success rate (for 50 problem instances) and average runtime (in seconds) on the 3D Complex map.

WF	1	2	4	10
A*+FE	8.82	59.60	62.05	62.59
A*+FBE	7.61	1.83	1.65	1.15
A*+E-FBE	**5.60**	**0.93**	**0.86**	**0.36**
MM+FE	7.72	212.55	289.27	300.00
MM+FBE	18.46	55.67	66.20	137.76
MM+E-FBE	7.99	5.55	11.45	29.84

Table 3: Average runtime (in sec) on random 3D maps.

$\frac{\epsilon_1}{\epsilon_2}$, assuming $\epsilon_1 \geq \epsilon_2$, and (2) *border density* (BD), which is the percentage of border locations on the separating area between the two domains. We consider WF=$\{1, 2, 4, 10\}$ with BD=20%, and BD=$\{5\%, 10\%, 40\%, 100\%\}$ with WF=4. We experimented on an AMD® EPYC 64 core-7702P @2.00GHz processor with 16GB of RAM.

First, we experimented on 50 problems instances of the 2D Cauldron map. To create large instances, we duplicated the 2D Cauldron map 16 times, as illustrated in Figure 2(a). The border locations are the purple vertical line between the two domains. We measured the average number of expansions (in millions) and average runtime (in seconds). Table 1 shows the results for this experiment. The first three rows present results for A* and the three other rows present results for MM. As can be seen, while A* with FBE and E-FBE had the lowest number of expansions, their heuristic calculation consumes time and A*+FE had almost the exact same runtime. Clearly, E-FBE runs faster than FBE for both A* and MM. WF has greater impact on the algorithms than BD. As mentioned, our A* search starts from the side of the search with the higher weight and, thus, when WF increases, fewer nodes are expanded on the side with the lower weight; however, the heuristic of FE becomes less informed. As MM searches from both sides, when WF increases, more nodes need to be expanded.

We performed a similar experiment on the 3D Complex map. The 3D Complex map was duplicated twice, one for each agent, and the border locations are located on the plane between the two domains (Figure 2(b)). Here, some of the algorithms could not solve some of the instances within a time limit of 5 minutes. We present, in Table 2, the number of instances each algorithm solved within the time limit (success rate) and the average runtime. The runtime of unsolved instances was set to the time limit (300 seconds) and the average runtime was computed out of all instances. While all solvers solved almost all problem instances, MM+FE was not able to solve many of the instances. Here also, WF influenced on the algorithms more than BD. While MM+FE solved all 50 instances with WF=1, only one instance was solved with WF=10. Here, both FBE and E-FBE ran much faster than FE for both A* and MM.

We also experimented on random 3D maps. We created two 200x200x200 cubes, one for each agent, connected on one side of each cube. We created 50 instances with BD=20%, WF=$\{1, 2, 4, 10\}$, and 10% random obstacles. Some instances were not solved by MM+FE and MM+FBE. Table 3 shows the average runtime for this experiment. Here too, E-FBE outperformed FBE, and FBE outperforms FE.

A* and MM can be seen as two extremes, namely, A* only searches from one side of the search and MM searches from both sides equally. Solving MATB is more beneficial when we progress more from the side with the larger weight. Thus, MM is less effective. Other Bi-HS algorithms, such as fMM (Shaham et al. 2017), have a parameter (a fraction) that defines how much each side progresses. Selecting the best fraction for a MATB instance is left for future work.

Conclusions

In this research, we studied MATB, where a meeting location at the border is required. We solved MATB with Uni-HS (A*) and Bi-HS (MM). We suggested a new type of heuristic function for MATB, called *Front-to-Border-to-End* (FBE)

and an enhanced method for calculating FBE (E-FBE). Our experiments show that FBE outperforms FE, and that E-FBE reduces the runtime. Future work may: (1) generalize MATB for multiple agents, for multiple tasks, or for continuous environments; (2) adjust advanced Bi-HS methods for solving MATB faster, such as *NBS* (Chen et al. 2017), *Bounds Propagation* (Shperberg et al. 2019a) and *DVCBS* (Shperberg et al. 2019b); or (3) improve E-FBE by passing border heuristics from a node to its successors.

Acknowledgments

This research was sponsored by the United States-Israel Binational Science Foundation (BSF) under grant numbers 2017692 and 2021643, and by Israel Science Foundation (ISF) under grant number 844/17.

References

Atzmon, D.; Freiman, S. I.; Epshtein, O.; Shichman, O.; and Felner, A. 2021. Conflict-Free Multi-Agent Meeting. In *the International Conference on Automated Planning and Scheduling (ICASP)*, 16–24.

Atzmon, D.; Li, J.; Felner, A.; Nachmani, E.; Shperberg, S. S.; Sturtevant, N.; and Koenig, S. 2020. Multi-Directional Heuristic Search. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 4062–4068.

Brewer, D.; and Sturtevant, N. R. 2018. Benchmarks for Pathfinding in 3D Voxel Space. In *the International Symposium on Combinatorial Search (SoCS)*, 143–147.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-End Bidirectional Heuristic Search with Near-Optimal Node Expansions. In *the International Joint Conference on Artificial Intelligence (IJCAI)*, 489–495.

Chen, R. 1984. Location problems with costs being sums of powers of Euclidean distances. *Computers & Operations Research*, 11(3): 285–294.

Cooper, L. 1968. An extension of the generalized Weber problem. *Journal of Regional Science*, 8(2): 181–197.

Dillenburg, J. F.; and Nelson, P. C. 1994. Perimeter search. *Artificial Intelligence*, 65: 165–178.

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*, SSC-4(2): 100–107.

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.

Izmirlioglu, Y.; Pehlivan, B. A.; Turp, M.; and Erdem, E. 2017. A general formal framework for multi-agent meeting problems. In *the IEEE International Conference on Robotics and Automation (ICRA)*, 1299–1306.

Lanthier, M. A.; Nussbaum, D.; and Wang, T.-J. 2005. Calculating the meeting point of scattered robots on weighted terrain surfaces. In *the Australasian Theory Symposium*, volume 41, 107–118.

Rosing, K. E. 1992. An optimal method for solving the (generalized) multi-Weber problem. *European Journal of Operational Research*, 58(3): 414–426.

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 *the International Symposium on Combinatorial Search (SoCS)*, 82–90.

Shperberg, S.; Felner, A.; Shimony, S.; Sturtevant, N.; and Hayoun, A. 2019a. Improving bidirectional heuristic search by bounds propagation. In *the International Symposium on Combinatorial Search (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 *the AAAI Conference on Artificial Intelligence (AAAI)*, 2379–2386.

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

Yan, D.; Zhao, Z.; and Ng, W. 2015. Efficient processing of optimal meeting point queries in Euclidean space and road networks. *Knowledge and Information Systems*, 42(2): 319–351.