Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

An Improved Meet in the Middle Algorithm for Graphs with Unit Costs

Edward C. Sewell,[1] John A. Pavlik,[2] Sheldon H. Jacobson[2]
[1]Dept. of Mathematics and Statistics,
Southern Illinois University Edwardsville, USA,
esewell@siue.edu
[2]Dept. of Computer Science,
University of Illinois, Urbana, USA,
jpavlik2@illinois.edu, shj@illinois.edu

Abstract

This paper proves several new properties of the Meet in the Middle (MMϵ) bidirectional heuristic search algorithm when applied to graphs with unit edge costs. Primarily, it is shown that the length of the first path discovered by MMϵ never exceeds the optimal length by more than one and that if the length of the first path found is odd, then it must be optimal. These properties suggest that the search strategy should emphasize finding a complete path as soon as possible. Computational experiments demonstrate that fully exploiting these new properties can decrease the number of nodes expanded by anywhere from twofold to over tenfold.

1 Introduction

Bidirectional Heuristic Search (BHS) algorithms essentially perform two interleaved A* searches, one in the forward direction and one in the backward direction, to find the lowest cost path from a given start node s to a given goal node t in a graph $G = (V, E)$. It is assumed throughout this paper that the cost of every edge is one, hence the lowest cost path is the shortest path as measured by the number of edges traversed. The unit cost assumption is valid for many types of problems to which bidirectional search has been applied. For example, the sliding tile problem and pancake puzzle both have unit costs and are tested in the computational results section of this paper. Generally the unit cost assumption is valid for any problem in which the objective is to find the number of nodes between the start and goal, in which case every edge naturally becomes unit cost.

BHS was introduced by Pohl (1971) with the goal of replacing one large unidirectional search by two much smaller searches, resulting in fewer node expansions in total. Despite a steady stream of research on BHS (see the review by Sturtevant and Felner (2018)), the results were largely disappointing. In many cases, BHS was outperformed by A* or its variants, and in the cases where BHS performed better than A*, the primary reason might have been better tie-breaking rules that lead to finding the shortest path sooner. These disappointing computational results were followed by a theoretical analysis (Kaindl and Kainz 1997) demonstrating that if all the f-values are distinct, then traditional BHS algorithms must expand at least as many nodes as either forward A* or reverse A* expands. More recently, Barker and Korf (2015) analyzed the viability of BHS algorithms compared to unidirectional search with an emphasis on the quality of the heuristic that is available for the problem. Their conclusion, with a few caveats, was: If a strong heuristic is available then either forward or reverse A* will outperform BHS, and if the heuristic is weak, then brute-force bidirectional search (i.e., no heuristic is used) will outperform BHS. Their analysis only applies to front-to-end BHS algorithms.

As part of Barker and Korf's analysis, they posited the existence of a BHS algorithm that is guaranteed to meet in the middle, although no such algorithm existed at that time. In response, Holte et al. (2016) developed the MM algorithm that is guaranteed to meet in the middle. Sharon et al. (2016) presented a variation of MM named MMϵ. Both algorithms are extensively analyzed by Holte et al. (2017).

This line of meet in the middle research has been extended in several directions. Shaham et al. (2017) presented Fractional MM (fMM), which is a generalization of MM that permits flexibility in choosing the meeting point of the forward and backward searches. Shaham et al. (2018) generalized the theory behind the fMM algorithm and constructed the Meet at the Threshold (MT) algorithm that uses a threshold parameter to control the meeting point of the forward and backward search. Eckerle et al. (2017) characterized pairs of nodes that must be expanded by any BHS algorithm that satisfies certain assumptions. This characterization served as the basis for the development of the Near-Optimal Bidirectional Search (NBS) front-to-end BHS algorithm (Chen et al. 2017). In this paper, they developed a method to determine the minimum number of nodes that must be expanded by any front-to-end BHS algorithm (under the same assumptions as in the paper by Eckerle et al.). They proved that NBS will never expand more than twice the number of such nodes and that no BHS algorithm can provide a stronger guarantee than this. Shperberg et al. (2019) constructed the Dynamic Vertex Cover Bidirectional Search (DVCBS) algorithm, which improves upon NBS in practice although it does not share NBS's theoretical guarantees. Finally, Barley et al. (2018) created an algorithm named Generalized Breadth-First Heuristic Search (GBFHS), which controls where the forward and backward searches meet by expanding one level of nodes at a time.

One conclusion that can be drawn from this line of research is that Barker and Korf's assessment of the viability of BHS was overly pessimistic. The papers above have demonstrated that BHS can outperform both A* and brute-force bidirectional search on a variety of problem domains and with heuristics of diverse strengths. Another conclusion that can be drawn is that BHS is an active area of research and it appears that there is still room for improvement. This paper continues this line of research by proving several new properties of the MMϵ algorithm when applied to graphs with unit edge costs. The primary result is the length of the first path discovered by MMϵ never exceeds the optimal length by more than one and that if the length of the first path found is odd, then it must be optimal. These properties suggest that the search strategy should emphasize finding a complete path as soon as possible. Computational experiments confirm that this approach can reduce the number of nodes expanded and the resulting algorithm is competitive with the best of the algorithms described in the previous paragraph.

2 New Properties of MMϵ for Graphs with Unit Costs

The GBFHS algorithm is designed such that if it is applied to a graph with unit costs, then the first path it discovers is an optimal path. MMϵ does not share this property, but in this section we prove that the first path discovered never exceeds the optimal length by more than one and that if the length of the first path is odd, then it must be optimal.

It is assumed that a path from s to t exists in order to simplify the presentation. It also is assumed that an admissible heuristic h_F (h_B) is available in the forward (backward) direction. Let $g_F(n)$ be the distance from s to node n, $f_F(n) = g_F(n) + h_F(n)$, C_F be the set of closed nodes in the forward direction, O_F be the set of open nodes in the forward direction, $gmin_F = \min\{g_F(n) : n \in O_F\}$, and $fmin_F = \min\{f_F(n) : n \in O_F\}$. For the backward search, $g_B(n)$, $f_B(n)$, C_B, O_B, $gmin_B$, and $fmin_B$ are defined analogously.

The particular algorithm of interest in this paper is the MMϵ algorithm. It defines the following *priority* functions:

$$p_F(n) = \max(2g_F(n) + \epsilon, f_F(n))$$

and

$$prmin_F = \min\{p_F(n) : n \in O_F\},$$

with $p_B(n)$ and $prmin_B$ defined analogously. The direction for each iteration is selected by choosing the direction with the smaller value of $prmin_F$ and $prmin_B$. Once a direction is selected, say the forward direction, a node with the minimum value of p_F is chosen as the next node to be expanded. It is these definitions that permit MMϵ to satisfy the following definition.

Definition 1. (Meet in the middle). *A bidirectional search algorithm meets in the middle if its forward search never expands a node n with $g_F(n) > C^*/2$ and its backward search never expands a node n with $g_B(n) > C^*/2$. A bidirectional search algorithm strongly meets in the middle if its forward search never expands a node n with $g_F(n) > (C^* - \epsilon)/2$ and its backward search never expands a node n with $g_B(n) > (C^* - \epsilon)/2$, where ϵ is the smallest cost of an edge in G.*

Algorithm 1: Pseudocode for MMϵ

```
1  $g_F(s) := g_B(t) = 0$
2  $O_F := \{s\}$
3  $O_B := \{t\}$
4  $U := \infty$
5  while ($O_F \neq \emptyset$) and ($O_B \neq \emptyset$) do
6      $C := \min(prmin_F, prmin_B)$
7      if $U \leq \max(C, fmin_F, fmin_B, gmin_F + gmin_B + \epsilon)$ then
8          return $U$
9      if $C = prmin_F$ then
10         // Expand in the forward direction
11         choose $n \in O_F$ for which $p_F = prmin_F$
12         move $n$ from $O_F$ to $C_F$
13         foreach child $c$ of $n$ do
14             if $c \in O_F \cup C_F$ and $g_F(c) \leq g_F(n) + cost(n,c)$ then
15                 continue
16             if $c \in O_F \cup C_F$ then
17                 remove $c$ from $O_F \cup C_F$
18             $g_F(c) := g_F(n) + cost(n,c)$
19             add $c$ to $O_F$
20             if $c \in O_B$ then
21                 $U := \min(U, g_F(c) + g_B(c))$
22     else
23         // Expand in the backward direction analogously
24 return $\infty$
```

If a BHS algorithm satisfies the strongly meets in the middle definition, it will be denoted as an SMMBHS algorithm. MM is an example of a BHS algorithm that meets in the middle and MMϵ is an example of an SMMBHS algorithm. For the sake of completeness, the pseudocode for MMϵ from Holte et al. (2017) is provided in Algorithm 1.

The meet in the middle definition explicitly places restrictions on which nodes can be expanded, or equivalently, which nodes can be in C_F and C_B. It also implicitly places restrictions on which nodes can be generated, or equivalently, which nodes can be in $C_F \cup O_F$ and $C_B \cup O_B$. These restrictions are set forth in the next lemma and corollary.

Lemma 1. *If an SMMBHS algorithm is applied to a graph with unit edge costs, then $g_F(n) \leq (C^* + 1)/2$ for every node $n \in C_F \cup O_F$ and $g_B(n) \leq (C^* + 1)/2$ for every node $n \in C_B \cup O_B$.*

Proof. Proof in the forward direction. Suppose $n \in C_F \cup$

O_F. The conclusion clearly is true if $n = s$, so suppose $n \neq s$. Then n must have been generated by some other node, say m. The strong meet in the middle property then implies

$$\begin{aligned} g_F(n) &= g_F(m) + 1 \\ &\leq (C^* - 1)/2 + 1 \\ &= (C^* + 1)/2. \end{aligned}$$

■

Corollary 1. *If an SMMBHS algorithm is applied to a graph with unit edge costs and C^* is even, then $g_F(n) \leq C^*/2$ for every node $n \in C_F \cup O_F$ and $g_B(n) \leq C^*/2$ for every node $n \in C_B \cup O_B$.*

Proof. Proof in the forward direction. Given that C^* is even and that $g_F(n)$ is an integer, it follows from Lemma 1 that $g_F(n) \leq C^*/2$. ■

Every time MMϵ generates a node, it checks if that node already has been generated in the opposite direction. If so, a path from s to t has been discovered and the length of that path can serve as an upper bound on the length of the shortest path. This leads to the definition that a path from s to t has been *discovered* if and only if every node on the path is in $C_F \cup O_F \cup C_B \cup O_B$ and at least one node on the path has been generated in both the forward and reverse directions, i.e., there is a node n on the path such that $n \in C_F \cup O_F$ and $n \in C_B \cup O_B$. The following theorem uses Lemma 1 to prove that the length of the first path discovered by MMϵ never exceeds C^* by more than one.

Theorem 1. *If an SMMBHS algorithm is applied to a graph with unit edge costs, then the length of the first path discovered by the algorithm is less than or equal to $C^* + 1$.*

Proof. Suppose the first path, say P, from s to t discovered by the algorithm has length d greater than $C^* + 1$.

Case 1. d is even. In this case, there is one node, say m, exactly in the middle of P, i.e., $g_F(m) = g_B(m) = d/2$. Thus,

$$g_F(m) = d/2 > (C^* + 1)/2.$$

Lemma 1 then implies that $m \notin C_F \cup O_F$. Similarly, $m \notin C_B \cup O_B$. This contradicts that P has been discovered.

Case 2. d is odd. In this case, there are two consecutive nodes, say m_1 and m_2, in the middle of P such that $g_F(m_1) = g_B(m_2) = (d-1)/2$. Then

$$\begin{aligned} g_B(m_1) &= g_B(m_2) + 1 = (d-1)/2 + 1 \\ &> (C^* + 1 - 1)/2 + 1 = (C^* + 2)/2. \end{aligned}$$

Lemma 1 then implies that $m_1 \notin C_B \cup O_B$. Similarly, $m_2 \notin C_F \cup O_F$. This leads to the conclusion that no node on P is in both $C_F \cup O_F$ and $C_B \cup O_B$, which contradicts that P has been discovered.

■

The following corollary uses Theorem 1 to conclude that the first path discovered by MMϵ is optimal whenever C^* is even.

Corollary 2. *If an SMMBHS algorithm is applied to a graph with unit edge costs and C^* is even, then the first path discovered by the algorithm is optimal.*

Proof. Suppose the first path, say P, from s to t discovered by the algorithm has length d greater than C^*. Theorem 1 then implies that $d = C^* + 1$, which in turn implies d is odd, because C^* is even. The proof of Case 2 of Theorem 1 can be modified slightly to show that $g_B(m_1) = (C^* + 2)/2$, leading to the same contradiction that P has not been discovered. ■

Theorem 1 and Corollary 2 can be combined to yield the following corollary.

Corollary 3. *If an SMMBHS algorithm is applied to a graph with unit edge costs and the length of the first path discovered is odd, then it is optimal.*

Proof. Suppose the first path, say P, from s to t discovered by the algorithm has odd length d. Corollary 2 states that if C^* is even, then the first path discovered is optimal, which means that the first path must have even length. Consequently, C^* cannot be even, hence it is odd. Theorem 1 then implies that $d \leq C^* + 1$, which in turn implies $d = C^*$ because d and C^* are both odd. ■

It is important to notice that while this paper focuses on improving the MMϵ algorithm, the results in this section apply to any BHS algorithm that satisfies the strongly meet in the middle property given in Definition 1.

3 Search Strategy

Broadly speaking, a search algorithm must perform two tasks, namely, it must find the optimal path and it must verify that this path is optimal by expanding nodes until the termination criteria is satisfied. One of the stopping criteria included in MMϵ is $U \leq gmin_F + gmin_B + \epsilon$, where U is the length of the shortest path found so far. To exploit this stopping criteria, the creators of MMϵ decided that whenever $prmin_F = prmin_B$ the algorithm should continue to expand nodes in the same direction until there no longer is a tie or $gmin$ in that direction has increased. Furthermore, they chose to break ties among nodes with the same priority value in favor of nodes with smaller g. Their search strategy was designed to focus on verifying optimality quickly, not on finding the optimal quickly.

The new properties enumerated in Theorem 1 and Corollaries 2 and 3 suggest that for graphs with unit costs, it might be better to use a search strategy that emphasizes finding the shortest path as soon as possible rather than trying to exploit the stopping criteria. After all, if the length of the first path discovered is odd, then the algorithm can be terminated immediately, without satisfying any of the other stopping criteria. The search strategy tested in this paper employs two phases corresponding to the two tasks that must be accomplished, i.e., finding and optimal path and verifying its optimality. The first phase breaks ties in choosing the direction and ties among nodes with the same priority with the goal of discovering a path as soon as possible. The second phase, which begins as soon as the first path has been discovered, breaks ties with the goal of verifying the optimality of the path that has been discovered. Of course, it may be the case that the first path discovered is not optimal, in which case the second phase can terminate as soon as it finds the lower cost path which is guaranteed to be optimal.

Algorithm 2: Pseudocode for choosing the direction

```
1  if $prmin_F < prmin_B$ then return $F$
2  if $prmin_F > prmin_B$ then return $B$
3  if $U = \infty$ then
4      // Phase I
5      if $\min\{f_F(n) : n \in O_F, p_F(n) = prmin_F\} \leq$
         $\min\{f_B(n) : n \in O_B, p_B(n) = prmin_B\}$ then
6          return $F$
7      else
8          return $B$
9  else
10     // Phase II
11     if $U$ or $prmin_F$ or $prmin_B$ changed during
         previous iteration then
12         if $|O_F| \leq |O_B|$ then return $F$
13         else return $B$
14     else
15         return direction used in previous iteration
```

In particular, the first phase breaks ties in favor of smaller values of $fmin$, i.e., whenever $prmin_F = prmin_B$ the direction containing the node with the smallest value of f is chosen, regardless of the direction chosen in the previous iteration. Furthermore, once a direction has been chosen, ties among nodes with the same priority value in that direction are broken in favor of nodes with smaller f. The second phase uses p-leveling, meaning that when a direction is chosen, all the nodes with minimum p-value are expanded in that direction before switching to the other direction. If $prmin_F = prmin_B$ when the first path is discovered, then the tie is broken by choosing the direction that has fewer open nodes. Within a given direction, ties are still broken in favor of nodes with smaller f with the goal of finding a shorter path, if one exists. The pseudocode for choosing the search direction is provided in Algorithm 2. The resulting algorithm is called Meet in the Middle for Unit Costs (MMUCϵ).

4 Computational Results

Computational experiments were conducted on the pancake problem and the 15-sliding tile puzzle.

For the pancake problem, one thousand random instances were generated for each value of $n = 10, 20, 30, 40$. The gap heuristic (Helmert 2010) was used. The strength of the heuristic was varied by using GAP-X, where the gaps involving the X smallest pancakes are ignored. The number of nodes that could be stored was limited to 300 million. The results are presented in Table 1. In this table, n is the number of pancakes and the columns labelled MMϵ and MMUCϵ contain the average number of nodes expanded by each of these algorithms. If the entry is a percentage, then this is the percentage of the 1,000 instances that were successively solved before running out of memory. For $n = 40$, MMϵ was only able to solve 969 of the instances before exceeding the memory limitation; the averages shown in Table 1 are limited to these 969 instances. The column labelled Ratio is the ratio of the number of nodes expanded by MMϵ to the number expanded by MMUCϵ. Table 1 shows that this ratio grows as the number of pancakes increases, culminating in a reduction factor of 15.9 for $n = 40$ with GAP-0. The same trend is seen for GAP-1. Only the smallest instances ($n = 10$) could be solved reliably when GAP-2 or GAP-3 was used. The ratio for these instances is smaller, perhaps because both algorithms are degenerating towards brute-force search due to the weaker heuristic.

n	MMϵ	MMUCϵ	Ratio
GAP			
10	76	40	1.9
20	6,905	1,099	6.3
30	127,634	11,738	10.9
40	1,312,112	82,558	15.9
GAP-1			
10	559	286	2.0
20	881,162	91,202	9.7
30	61.4%	91.2%	
GAP-2			
10	1,681	1,105	1.5
20	56.6%	95.0%	
GAP-3			
10	2,433	2,089	1.2

Table 1: Computational results for the pancake problem.

MMϵ	MMUCϵ	Ratio
1,634,245	687,669	2.4

Table 2: Computational results for the 15-puzzle.

For the 15-puzzle, the algorithms were tested on the 100 problems created by Korf (1985). A 3-4-4-4 disjoint pattern database (Korf and Felner 2002) was used as the heuristic. The stronger 7-8 database was not used because it takes longer to generate the 7-8 database than to solve all 100 problems using the 3-4-4-4 database. The sliding tile puzzle has the property that every path from the starting configuration to the goal configuration has the same parity. In this case, Theorem 1 and Corollary 2 imply that the first path discovered by MMϵ must be optimal. On average, MMUCϵ reduced the number of nodes expanded by a factor of 2.4.

Table 3 compares MMUCϵ to MMϵ and NBS, as presented in Chen et al. (2017), on a set of 50 randomly generated pancake problems with 16 pancakes and on the 100 test problems for the 15-puzzle problem using the Manhattan Distance (MD) heuristic. Table 3 also includes the results obtained by GBFHS, as presented in Barley et al. (2018), on the same sets of instances.

Table 3 shows that MMUCϵ significantly outperformed MMϵ and NBS on the pancake instances when GAP-2 and GAP-3 were used. MMUCϵ performed slightly better than MMϵ and NBS on the 15-puzzle instances. MMUCϵ performed slightly worse than GBFHS on the pancake in-

h_F, h_B	MMϵ	NBS	GBFHS	MMUCϵ
GAP	283	335	279	315
GAP-2	587,283	625,900	250,941	293,874
GAP-3	7,100,998	6,682,497	2,140,718	2,454,084
MD	13,162,312	12,851,889	12,507,393	12,270,697

Table 3: Computational comparison of MMUCϵ to MMϵ, NBS, and GBFHS for the pancake problem and the 15-puzzle.

stances and slightly better on the 15-puzzle instances. We just learned of the DVCBS algorithm and are in the process of obtaining the test instances that were used in that paper.

5 Conclusions

We have established a stronger termination condition for SMMBHS algorithms such as MMϵ when applied to the unit cost domain. That analysis lead us to implement a two-phase search strategy which resulted in the new MMUCϵ algorithm as a modification of MMϵ. The new MMUCϵ algorithm is competitive with the best BHS algorithms that have been published.

References

Barker, J. K., and Korf, R. E. 2015. Limitations of front-to-end bidirectional heuristic search. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 1086–1092.

Barley, M.; Riddle, P.; López, C. L.; Dobson, S.; and Pohl, I. 2018. Gbfhs: A generalized breadth-first heuristic search algorithm. In *Proceedings of the Eleventh International Symposium on Combinatorial Search (SoCS 2018)*, 28–36.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. 2017. Front-to-end bidirectional heuristic search with near-optimal node expansions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-2017)*, 489–495.

Eckerle, J.; Chen, J.; Sturtevant, N.; Zilles, S.; and Holte, R. C. 2017. Sufficient conditions for node expansion in bidirectional heuristic search. In *International Conference on Automated Planning and Scheduling (ICAPS-2017)*, 79–87.

Helmert, M. 2010. Landmark heuristics for the pancake problem. In Felner, A., and Sturtevant, N., eds., *Proceedings of the Third Annual Symposium on Combinatorial Search (SOCS-2010)*, 109–110. AAAI Press.

Holte, R. C.; Felner, A.; Sharon, G.; and Sturtevant, N. R. 2016. Bidirectional search that is guaranteed to meet in the middle. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*.

Holte, R. C.; Felner, A.; Sharon, G.; Sturtevant, N. R.; and Chen, J. 2017. MM: A bidirectional search that is guaranteed to meet in the middle. *Artificial Intelligence* 252:232–266.

Kaindl, H., and Kainz, G. 1997. Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research* 7:283–317.

Korf, R. E., and Felner, A. 2002. Disjoint pattern database heuristics. *Artificial Intelligence* 134:9–22.

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

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

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 *Proceedings of the Tenth International Symposium on Combinatorial Search (SoCS 2017)*, 82–90.

Shaham, E.; Felner, A.; Sturtevant, N. R.; and Rosenschein, J. S. 2018. Minimizing node expansions in bidirectional search with consistent heuristics. In *Proceedings of the Eleventh International Symposium on Combinatorial Search (SoCS 2018)*, 81–89.

Sharon, G.; Holte, R. C.; Felner, A.; and Sturtevant, N. R. 2016. Extended abstract: An improved priority function for bidirectional heuristic search. In *Proceedings of the Ninth International Symposium on Combinatorial Search (SoCS 2016)*, 139–140.

Shperberg, S. S.; Felner, A.; Sturtevant, N. R.; Shimony, S. E.; and Hayoun, A. 2019. Enriching non-parametric bidirectional search algorithms. In *Proceeding of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

Sturtevant, N. R., and Felner, A. 2018. A brief history and recent achievements in bidirectional search. In *Proceeding of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 8000–8008.