The Minimal Set of States that Must be Expanded in a Front-to-end Bidirectional Search

Eshed Shaham
School of Engineering and CS
Hebrew University of Jerusalem
Jerusalem, Israel
eshed.shaham@mail.huji.ac.il

Ariel Felner
ISE Department
Ben-Gurion University
Be'er-Sheva, Israel
felner@bgu.ac.il

Jingwei Chen
Computer Science Department
University of Denver
USA
chenjingwei1991@gmail.com

Nathan R. Sturtevant
Computer Science Department
University of Denver
USA
sturtevant@cs.du.edu

Abstract

A* is optimal among admissible unidirectional algorithms when searching with a consistent heuristic. Recently, similar optimality bounds have been established for bidirectional search but, no practical algorithm is guaranteed to always achieve this bound. In this paper we study the nature of the number of nodes that must be expanded in any front-to-end bidirectional search. We present an efficient algorithm for computing that number and show that a theoretical parameterized generalization of MM, with the correct parameter, is the optimal front-to-end bidirectional search. We then experimentally compare various algorithms and show how far they are from optimal.

1 Introduction and Overview

What is the best possible algorithm that one could use to solve a heuristic search problem? A* running on a problem instance with a consistent heuristic is known as *optimally effective* in terms of necessary node expansions (those with $f < C^*$) for unidirectional search algorithms.

Recent theoretical work describes which states must be expanded by an admissible front-to-end bidirectional heuristic search algorithm with a consistent heuristic (Eckerle et al. 2017). In bidirectional search the conditions for expansions apply to pairs of states, and for every eligible pair, only one of the two states must be expanded. Chen et al. (2017) show that the minimal set of nodes that must be expanded corresponds to the minimal vertex cover of a *Must-Expand Graph* (G_{MX}), a bipartite graph whose edges correspond to states pairs. The minimum vertex cover of G_{MX}, denoted hereafter by VC, establishes the minimum number of node expansions necessary to prove the optimal solution by any search algorithm, unidirectional or bidirectional. The recently introduced *Near Optimal Bidirectional Search* (NBS) (Chen et al. 2017) expands at most 2·VC nodes. Furthermore, no algorithm can do better than NBS in the worst case.

In this paper we introduce an efficient algorithm for computing VC. We also introduce a new bidirectional search algorithm called *Fractional MM* (fMM) which is a generalization of MM (Holte et al. 2016). MM is a bidirectional search algorithm which guarantees that the search frontiers meet in the middle. fMM has a parameter $0 \leq p \leq 1$ that allows a more flexible meeting point. We prove that there exists a fraction p^* such that fMM(p^*) is optimally effective. This algorithm is primarily of theoretical interest, since p^* is not known *a priori*. But, we use fMM to study the optimal meeting point in different benchmark domains and to compare the suboptimality of NBS, A*, and MM.

2 Terminology and Notation

A problem instance is a pair $(start, goal)$ of states in a state-space G with non-negative edge weights. The aim of search is to find a *least-cost path* from $start$ to $goal$. $d(u, v)$ is the distance (cost of a least-cost path) from state u to state v. $C^* = d(start, goal)$ is the cost of an optimal solution.

Front-to-end bidirectional search algorithms interleave two separate searches, a search forward from the start state, and a search backward from the goal state. We use f_F, g_F and h_F to indicate costs of the forward search and f_B, g_B and h_B to indicate costs in the backward search. Likewise, $Open_F$ and $Open_B$ store states generated in the forward and backward directions, respectively.

Front-to-end algorithms use two heuristic functions. The *forward heuristic*, h_F, is *forward admissible* iff $h_F(u) \leq d(u, goal)$ for all u in G and is *forward consistent* iff $h_F(u) \leq d(u, u') + h_F(u')$ for all u and u' in G. The *backward heuristic*, h_B, is *backward admissible* iff $h_B(v) \leq d(start, v)$ for all v in G and is *backward consistent* iff $h_B(v) \leq d(v', v) + h_B(v')$ for all v and v' in G.[1]

A problem instance is solvable if there is a forward path from $start$ to $goal$. I_{AD} is the set of solvable problem instances in which h_F is forward admissible and h_B is backward admissible. I_{CON} is the subset of I_{AD} in which h_F is forward consistent and h_B is backward consistent. A search algorithm is *admissible* iff it is guaranteed to return an optimal solution for any problem instance in I_{AD}.

We assume that search algorithms only have black-box access to the expand, heuristic, and cost functions. This follows the assumptions of Dechter and Pearl (1985) formal-

[1] Front-to-front algorithms use a heuristic $h(u, v)$ defined on any two states $(u, v) \in G$ and may perform heuristic lookup between any two states that belong to opposite frontiers. The theory here does not apply to front-to-front algorithms.

ized in the description of *Deterministic, Expansion-based Black-box (DXBB)* algorithms (Eckerle et al. 2017). DXBB algorithms have no a priori information about the instance(s) they are going to solve, nor do they have any knowledge of the heuristic besides that it is admissible.

3 Background

Next we cover recent bidirectional algorithms and the theory that forms the basis of the contributions of this paper.

3.1 The Meet in the Middle algorithm (MM)

MM (Holte et al. 2016) is the first bidirectional heuristic search algorithm guaranteed to "meet in the middle". That is, MM will never expand a state whose g-value exceeds $C^*/2$. Assuming that the gray ovals of Figure 2 (middle) represent the states expanded by MM, these ovals are guaranteed to stay inside the enclosing circles. As a result, the frontiers must meet at the midpoint of the optimal solution.

This is achieved by MM's novel priority functions. For the forward direction $pr_F(u) = \max(f_F(u), 2g_F(u))$ and for the backward direction $pr_B(u) = \max(f_B(u), 2g_B(u))$.

MM chooses to expand a node with minimum priority from either direction (labeled C). The $2g_F(u)$/$2g_B(u)$ terms in the priority function are used to ensure that the searches meet in the middle (property **P1** in (Holte et al. 2016)).

MM keeps track of L, the cheapest path from $start$ to $goal$ seen so far, using immediate solution detection (Sturtevant and Chen 2016) to check for a solution when new states are generated. MM can terminate with the optimal solution when $L \leq \max(C, fmin_F, fmin_B, gmin_F+gmin_B+\epsilon)$, where $fmin$ and $gmin$ are the minimal f- and g-values on the respective open lists and ϵ is the cost of the cheapest edge in the state-space if it is known. Therefore, MM can safely stop if L is *smaller than or equal to* any of the lower-bound terms inside the max.

3.2 Sufficient Conditions for Node Expansion

In a unidirectional search, all states with $f_F(s) < C^*$ are necessarily expanded by any admissible algorithm running on I_{CON} (Dechter and Pearl 1985). Thus, A* is *optimally effective* in terms of state expansions. In bidirectional search this property does not hold.

Recent theoretical work (Eckerle et al. 2017) generalized this theory to bidirectional search. In bidirectional search, necessary expansions are defined in terms of pairs of states, one in the forward direction and one in the backward direction. A state pair (u, v) has been *expanded* if either u has been expanded in the forward direction or v has been expanded in the backward direction (we do not require both). Similarly, the pair (u, v) is a *must-expand pair* if (u, v) must be expanded (i.e., either u in the forward direction or v in the backward direction) by an algorithm that is guaranteed to return an optimal solution.

Eckerle et al. (2017) showed that any admissible bidirectional search algorithm searching with a consistent heuristic must expand the state pair (u, v) if the following three conditions are met: (1) $f_F(u) < C^*$, (2) $f_B(v) < C^*$, and (3) $g_F(u) + g_B(v) < C^*$. The pair (u, v) is then called a *must-expand pair*. If neither u nor v is expanded there might be a path shorter than C^* that connects u and v. A more formal statement and proof of the theorem (which is proven using *path pairs* instead of state pairs) is found in Theorem 1 of Eckerle et al. (2017).

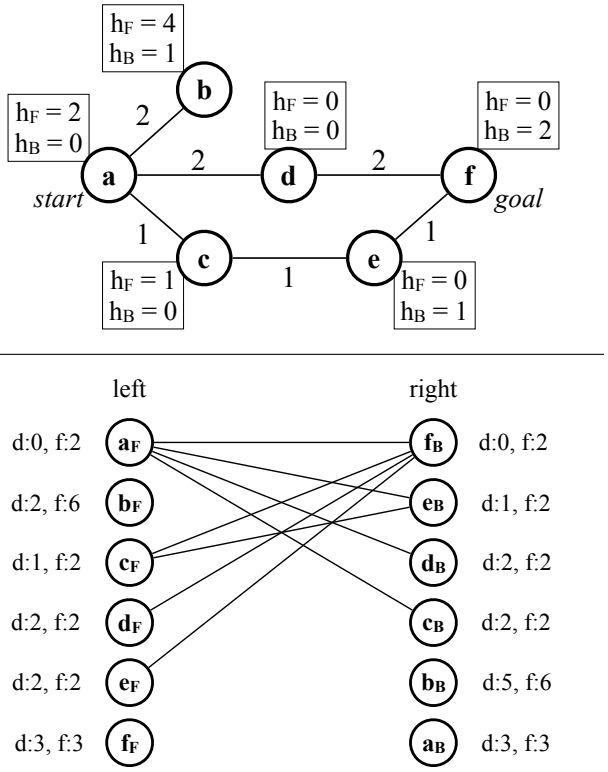


Figure 1: A sample graph (top) and G_{MX} (bottom)

3.3 The *Must-Expand Graph* G_{MX}

The conditions for state-pair expansion define a bipartite graph called the *Must-Expand Graph* ($G_{MX}(I)$) (Chen et al. 2017).

Definition 1. *For each pair of states* (u, v) *define*

$$lb(u, v) = \max\{f_F(u), f_B(v), g_F(u) + g_B(v)\}$$

When h_F is forward admissible and h_B is backward admissible, $lb(u, v)$ is a lower bound on the cost of a path from $start$ through u, connecting to v, and then to $goal$.

Definition 2. *The Must-Expand Graph* $G_{MX}(I)$ *of problem instance* $I \in I_{CON}$ *is an undirected, unweighed bipartite graph defined as follows. For each state* $u \in G$*, there are two vertices in* $G_{MX}(I)$*, the left vertex* u_F *and right vertex* u_B*. For each pair of states* $u, v \in G$*, there is an edge in* $G_{MX}(I)$ *between* u_F *and* v_B *if and only if* $lb(u_F, v_B) < C^*$*. Thus, there is an edge in* $G_{MX}(I)$ *between* u_F *and* v_B *if and only if the pair* (u, v) *is a must-expand pair.*

It follows (see Chen et al. (2017)) that any algorithm that finds an optimal solution must expand nodes that belong to

a vertex cover of $G_{MX}(I)$. Figure 1 (top) shows a problem instance $I = (G, h_F, h_B) \in I_{CON}$. In this example a is the start state, f is the goal, and $C^* = 3$. Figure 1 (bottom), duplicated from Chen et al. (2017), shows $G_{MX}(I)$, where d refers to the cost of the shortest path to each state and f refers to the f-cost of that path. By construction, the edges in $G_{MX}(I)$ exactly correspond to the state pairs that must be expanded, and therefore any vertex cover for $G_{MX}(I)$ will, by definition, represent a set of expansions that covers all the required state pairs. For example, one possible vertex cover includes exactly the vertices in the left side with at least one edge: $\{a_F, c_F, d_F, e_F\}$. This represents expanding all the required state pairs in the forward direction. This requires four expansions and is not optimal because the required state pairs can be covered with just three expansions: a and c in the forward direction and f in the backward direction. This corresponds to a minimum vertex cover of $G_{MX}(I)$: $\{a_F, c_F, f_B\}$.

Throughout this paper VC is used to denote the *minimal vertex cover* of $G_{MX}(I)$. |VC| is a lower bound on the number of states that must be expanded. Chen et al. (2017) showed that for every DXBB algorithm there exists a worst-case graph where it will expand twice as many nodes as are in VC. Therefore, unlike unidirectional search, no DXBB bidirectional search algorithm can be guaranteed to be optimal in terms of state expansions.

3.4 NBS

Chen et al. (2017) introduced the Near-optimal Bidirectional Search algorithm (NBS), which efficiently finds a near-optimal vertex cover of $G_{MX}(I)$ and thus is near-optimal in terms of necessary node expansions. NBS chooses a pair of states (u, v) from $G_{MX}(I)$ with minimal $lb(u, v)$. While only one of these states must be included in VC, NBS expands both of them. This approach results in NBS expanding at most 2|VC| states.

In the next sections we study the structure of VC, providing an algorithm that can find VC in time linear in $|G_{MX}|$. This will enable us to measure how many extra nodes are expanded by various algorithms such as A*, MM and NBS when compared to |VC|. We will also define a parameterized algorithm, Fractional MM(p) (fMM(p)) and show that there exists p^* such that fMM(p^*) expands exactly the nodes in VC. However, fMM(p) is not a DXBB algorithm because the exact value for p^* is not known in advance.

4 Assumptions

The classic claim by Dechter and Pearl (1985) that all the nodes with $f < C^*$ must be expanded is for any admissible unidirectional algorithm. That is, the algorithm must be able to solve any instance in I_{AD} and may not assume that the heuristic is consistent. But, the claim only describes performance on problem instances with consistent heuristics (I_{CON}). In addition, nodes with $f = C^*$ are not included in the theory; algorithms might expand many or few such nodes depending on the problem instance and the tie-breaking rule. The bidirectional theory (Eckerle et al. 2017; Chen et al. 2017) has similar assumptions that also apply to our work. We list them here.

Assumption 1: Our proof of optimality below is only with respect to state pairs that *must be expanded*, i.e., we only consider states or pair of states with a lower bound that is strictly smaller than C^*. Nevertheless, in practice, depending on the tie breaking rule, NBS, MM, A* and any other algorithm may expand extra states or state pairs with $lb(U, V) = C^*$, but such states are not *must-expand* states. In the best case, there is only one such state – the goal state (in a non-pathological instance, see (Dechter and Pearl 1985)).

Assumption 2: We assume that our front-to-end algorithms are only coupled with forward and backward admissible heuristics but that no other information on the graph or on the heuristic is known to these algorithms. Therefore, these algorithms cannot assume that their heuristic is consistent and cannot use any form of front-to-front heuristics. Using such front-to-front heuristics is a subject of future work.

5 Fractional Meet in the Middle

Meeting in the middle, as achieved by MM, is not so much about where the meeting point is ($C^*/2$ for MM). A more important attribute is the guarantee that the forward/backwards searches will not venture beyond their respective distances from the start/goal to the meeting point. Previous bidirectional heuristic search algorithms did not have this guarantee. Their searches could expand many nodes beyond the meeting point as shown in the dotted shapes of Figure 2 (middle).

Definition 3. *Assume start and goal states for which $d(start, goal) = C^*$. Let S_F be a connected set of states which includes start and let S_B be a connected set of states which includes goal. We say that the pair of sets (S_F, S_B) is* **restrained** *if there exists a fraction $0 \leq p \leq 1$ such that S_F only includes states u with $g_F(u) \leq pC^*$ and S_B only includes states v with $g_B(v) \leq (1-p)C^*$.*

A bidirectional search algorithm is *restrained* if the states it expands are in a restrained pair of sets (S_F, S_B). $(pC^*, (1-p)C^*)$ is the *meeting point* of such an algorithm. MM is a special case of a *restrained* algorithm where $p = 1/2$. Similarly, A* and reverse A* (from the goal to start) are special cases where $p = 1$ and $p = 0$, respectively.

We now propose a generalization of MM called *Fractional MM* (fMM) which is *restrained*, but its meeting point can be any fraction p of C^*. fMM uses the same stopping condition of MM (described above) but it generalizes MM by using the following priority functions on paths in the open lists:

$$pr_F(u) = \max(f_F(u), g_F(u)/p)$$
$$pr_B(v) = \max(f_B(v), g_B(v)/(1-p)).$$

where $0 \leq p \leq 1$. We call this algorithm fMM(p).

fMM's forward and backward searches meet at $(pC^*, (1-p)C^*)$. That is, fMM(p) is restrained with respect to p. Neither the forward/backward search will expand states farther from the start/goal than the meeting point, as shown in Figure 2 (right). This claim is a generalization of property P1 from Holte et al. (2016), which claims that MM meets at $C^*/2$. However, fMM has flexibility of selecting where that

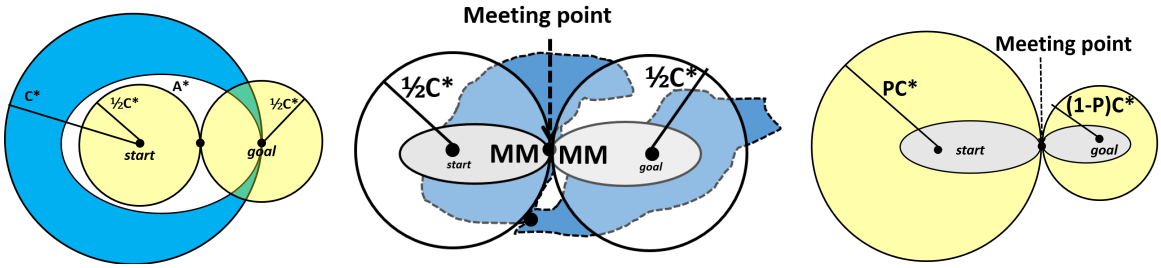


Figure 2: Left: A* vs. unidirectional/bidirectional brute-force search. Middle: bidirectional heuristic search and MM. Right: fMM.

meeting point will be. We will prove this claim for fMM with a slightly stronger condition in the next section.

The performance of fMM depends on the choice of p. In the best case the sum of the two search trees will be as small as possible. In some cases we can analyze the best choice of p. For example, let b_F and b_B be uniform and fixed branching factors of the forward and backward directions. It can be shown that if we set $p = \frac{log(b_F)}{log(b_F)+log(b_B)}$ then the two search trees will meet at $(pC^*, (1-p)C^*)$ with equal size search trees. In general, however, we don't have a general way to *a priori* choose p, but we will prove below that there exists a fraction p^* such that fMM(p^*) is optimally effective.

5.1 Fractional MM with Tie-Breaking

We next define a stronger version of restrained called *strongly restrained* and then a tie-breaking rule for fMM that which causes it to be strongly restrained. Below, we will prove that this new variant of fMM is optimally effective.

Definition 4. *The pair of sets (S_F, S_B) is* **strongly restrained** *if there exists a fraction $0 \leq p \leq 1$ such that S_F only includes states u with $g_F(u) < pC^*$ and S_B only includes states v with $g_B(v) < (1-p)C^*$. As above, this can also describe bidirectional search algorithms.*

Observation: States u with $g_F(u) = pC^*$ are allowed for *restrained* sets but not allowed for *strongly restrained* sets.

Next, we define a tie-breaking rule (**TB1**) for fMM and hereafter assume that TB1 is always applied for fMM.

TB1: if two nodes x and y (either on the same search direction or in opposite directions) have $pr(x) = pr(y)$ where $pr(x) = pr_F(x) = g_F(x)/p$ or $pr(x) = pr_B(x) = g_B(x)/(1-p)$ but $pr(y) = pr_F(y) \neq g_F(x)/p$ or $pr(y) = pr_B(y) \neq g_B(x)/(1-p)$ then choose to expand y.

This rule means that we prefer to expand nodes whose priority was solely determined by their f-values over nodes whose priority was determined by $g_F(x)/p$ or by $g_B(x)/(1-p)$ (either solely or with a tie to the f-value).

Theorem 1. fMM *with TB1 is strongly restrained. (Corresponding to P1 for MM.)*

Proof for the forward side: Let x be a state with $g_F(x) = pC^*$. We want to show that fMM(p) will never expand x. If $h_F(x) > (1-p)C^*$ then $pr_F(x) = g_F(x) + h_F(x) > C^*$ and x will never be expanded. If $h_F(x) \leq (1-p)C^*$ then $f_F(x) \leq C^*$ but $pr_F(x) = g_F(x)/p = C^*$ (the tie breaking rule will prefer to expand other nodes before x). Now, assume that we reach a stage where we are expanding states with $pr = C^*$. Let y be a state on the optimal path. If $pr_F(y) < C^*$ then y was already expanded. If $pr_F(y) = C^*$ and $g_F(y) < pC^*$ then y will be expanded before x due to the tie breaking rule. If $pr_F(y) = C^*$ and $g_F(y) = pC^*$ then due to the tie breaking rule, all states on the optimal path with $g_F < pC^*$ were forward expanded and y is in $Open_F$. Likewise, all states on the optimal path with $g_B < (1-p)C^*$ were backward expanded and y is in $Open_B$ as well. Thus, an optimal solution via y is found and the search halts before expanding x. □

This implies a version of MM which is strongly restrained. Holte et al. (2016) identified an area in the state space called NN (Near-Near). Any state x in NN has $g_F(x) = g_B(x) = C^*/2$. While MM may expand nodes in NN, MM with TB1 will never expand nodes in NN.

6 The Optimal Front-to-End algorithm

In this section we study the nature of the optimally effective front-to-end algorithm which expands no more nodes with $f < C^*$ than what must be expanded. To do this, we first show how to find the *minimal vertex cover* of G_{MX} (VC) and prove that it will be strongly restrained with respect to some $0 \leq p^* \leq 1$. Then, we describe an algorithm that will efficiently find p^* and compute the size of VC given C^* and the g-costs of vertices in G_{MX}. Finally, we will show that fMM(p^*) is optimally effective, i.e., that it only expands nodes with $f < C^*$ that belong to VC.

This study is theoretical in nature because G_{MX} and C^* are not known a priori and thus VC, and as a consequence p^*, cannot be known a priori. Thus, fMM(p^*) is not a DXBB algorithm. This aligns with the claim that no DXBB algorithm is optimal (Chen et al. 2017). But VC and fMM(p^*) can be used post priori for theoretical studies such as comparing the nodes expanded by an algorithm to the number of nodes that must be expanded.

6.1 Finding VC

How do we find VC? In general, minimal vertex cover is NP-complete, but G_{MX} is a bipartite graph for which there exists a polynomial algorithm that finds its minimal vertex cover with complexity $O(E\sqrt{V})$ (Hopcroft and Karp 1973). Note that G_{MX} may not be sparse, even if the state space from which it was derived is sparse. Furthermore, G_{MX} may

grow exponentially with the problem input size. Thus, even generating G_{MX} may exceed our computational resources, let alone finding its minimal vertex cover with this approach.

Below we show that states in G_{MX} can be clustered together into groups based on their g-values, and that the problem now becomes to find the minimal vertex cover of a graph $\hat{G}$ (see definition 6 below) that is based on these groups. Then, we show that, given knowledge of C^*, the minimal vertex cover of $\hat{G}$ is strongly restrained. Following that we provide an algorithm that finds the optimal value for p^* that defined the minimal vertex cover of $\hat{G}$ in time linear in the number of unique g-costs. Based on that, we show that $\texttt{fMM}(p^*)$ is optimally effective.

6.2 The Abstract Graph ($\hat{G}$)

Assume that both C^* and G_{MX} are given as input. We cluster together states with equal g-value on either side of G_{MX} as follows.

Definition 5. *$G_F(i)$ is the set of all states $x \in G_{MX}$ such that $g_F(x) = i$. Likewise, $G_B(j)$ is defined by $g_B(x) = j$.*

Note that *all* states in $G_F(i)$ have $f_F < C^*$ and *all* states in $G_B(j)$ have $f_B < C^*$ because this is a requirement for being included in G_{MX} (see definition 2). Therefore, any pair of states (u, v) $(u \in G_F(i), v \in G_B(j))$ is a *must-expand pair* if the third condition of definition 1, $g_F(u) + g_B(v) < C^*$, also applies. This will be true if $i+j < C^*$. We therefore define $(G_F(i), G_B(j))$ to be a *must-expand group pair* (denoted MEGP) if $i + j < C^*$. We denote the number of states in $G_F(i)$ by $N(G_F(i))$ and the number of states in $G_B(j)$ by $N(G_B(j))$. Observe that if $(G_F(i), G_B(j))$ is a MEGP then there are $N(G_F(i)) \times N(G_B(j))$ distinct must-expand pairs in these groups (= number of corresponding edges in G_{MX}).

Definition 6. *Let $\hat{G}$, the* **must-expand groups graph**, *be a bipartite graph which abstracts G_{MX} as follows. $\hat{G} = ((V_F, V_B), E)$. V_F includes a node for each nonempty $G_F(i)$ group. Similarly, V_B includes a node for each nonempty $G_B(j)$ group. Every pair of groups $(G_F(i), G_B(j))$ that are MEGP (for which $i + j < C^*$) induce an edge in $\hat{G}$. An edge $(G_F(i), G_B(j))$ represents the fact that each pair of states $u \in G_F(i)$ and $v \in G_B(j)$ is a* must-expand pair.

Figure 3 shows $\hat{G}$ for a unit-cost state space with $C^* = 4$.

6.3 Theoretical Analysis

We next prove a number of lemmas about $\hat{G}$ that identify VC for G_{MX}. For simplicity of terminology, when referring to $\hat{G}$, we will talk about groups $G_F(i)$ and $G_B(j)$ in $\hat{G}$ as shorthand for the nodes in $\hat{G}$ representing $G_F(i)$ and $G_B(j)$.

Lemma 1: In $\hat{G}$, group $G_F(i)$ is connected to all groups $G_B(j)$ for $j < C^* - i$, but $G_F(i)$ is not connected to nodes representing groups $G_B(j)$ for $j \geq C^* - i$.

Proof: Only groups $G_B(j)$ for which $j < C^* - i$ form a MEGP with $G_F(i)$ as $i + j < C^*$ (from definition 6). For example, as can be seen in Figure 3 group $G_F(1)$ is connected to groups $G_B(0)$, $G_B(1)$ and $G_B(2)$, but not to group $G_B(3)$. □

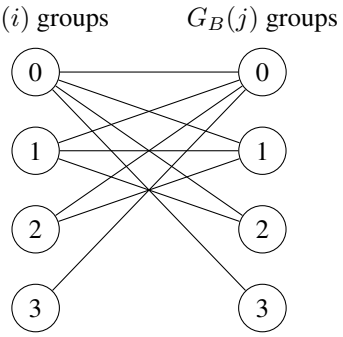


Figure 3: The graph $\hat{G}$ assuming $C^* = 4$ and unit edge costs. Numbers inside the nodes are the i or the j g-values.

Lemma 2: In G_{MX}, if $u \in G_F(i)$ belongs to VC then *all* other states in $G_F(i)$ must also belong to VC. Similarly, if $v \in G_B(j)$ belongs to VC then *all* other states in $G_B(j)$ must also belong to VC.

Proof: Assume the contrary, that $u \in G_F(i)$ is in VC but there exists $u' \in G_F(i)$ such that $u' \notin$ VC. Since $u \in$ VC, it is connected in G_{MX} to some state $v \in G_B(j)$ for some j. But, since all states in $G_B(j)$ have g-values equal to $g_B(v)$ then u is connected to *all* states in $G_B(j)$. Similarly, u' is also connected to all states in $G_B(j)$. But, since $u' \notin$ VC it means that *all* states in $G_B(j)$ must be in VC. This is true for all groups $G_B(j)$ connected to $G_F(i)$. So, u may be deleted from VC as all its neighbors are in VC, contradicting the assumption that VC is a minimum cover. □

Lemma 3: Considering $G_F(i)$ in G_{MX}, either all states $v \in G_F(i)$ are in VC or none of them are in VC.

Proof: This is a direct result of lemma 2. □

We now further extend $\hat{G}$ by adding weights to the nodes. In $\hat{G}$ the weight of node $G_F(i)$ is $N(G_F(i))$ and the weight of node $G_B(j)$ is $N(G_B(j))$. On such graphs a *minimal weighted vertex cover* is defined as follows.

Definition 7. *The minimal weighted vertex cover of a graph $G = (V, E, w)$ where vertices (not edges) have weights (i.e., $w : V \Rightarrow \mathbb{R}^+$) is a set of nodes $C \subseteq V$ such that every edge of G has at least one endpoint in C while the total accumulated weights of vertices in C is minimized.*

Recall that VC denotes the minimal vertex cover of G_{MX}. We hereafter use WVC to denote the minimal weighted vertex cover of $\hat{G}$.

Lemma 4: (WVC= VC) Let S be the set of nodes that form a WVC in $\hat{G}$. The states in G_{MX} that make up the groups in S are exactly those that form VC.

Proof: This is a direct result of the way $\hat{G}$ is built and our previous lemmas. In particular, VC only includes entire groups (Lemma 3) and it must cover all the edges of $\hat{G}$ as each such edges connect a MEGP. □

6.4 Attributes of WVC

Our next lemmas prove more attributes of WVC that will then show that VC is strongly restrained.

Lemma 5: In $\hat{G}$ for any $i' > i$, any group $G_B(j)$ that is connected to $G_F(i')$, is also connected to $G_F(i)$.

Proof: By the way $\hat{G}$ is built. Since $i' > i$ then if $i' + j < C^*$ it must be that $i + j < C^*$. For example, in Figure 3, group $G_B(1)$ is connected to $G_F(2)$ and to all the groups with smaller g_F-values, i.e., to $G_F(1)$ and to $G_F(0)$. □

Lemma 6: In $\hat{G}$, let $\underline{i}$ be the minimal g_F-value for which $G_F(\underline{i}) \notin WVC$. For every $i > \underline{i}$, $\quad G_F(i) \notin WVC$.

Proof: Since $G_F(\underline{i}) \notin$ WVC, all the groups $G_B(j)$ that are connected to $G_F(\underline{i})$ must be in WVC. But, by lemma 5, all groups that are connected to $G_F(i)$ must also be connected to $G_F(\underline{i})$ (as $i > \underline{i}$) and thus all these groups must be in WVC. Thus, $G_F(i) \notin WVC$. □

Lemma 7: (no gaps) WVC contains a series of consecutive groups (with no gaps in the middle). There exists a g_F-value $\underline{i} \geq 0$ such that WVC consists of *all* groups $G_F(i)$ for which $i < \underline{i}$. Likewise, there exists a g_B-value $\underline{j}$ such that WVC consists of *all* groups $G_B(j)$ for which $j < \underline{j}$.

Proof: A direct result of lemma 6.[2] □

Middle summary: Lemma 7 has shown that WVC is a series of consecutive groups with g_F-values smaller than $\underline{i}$ in the forward side and a series of consecutive groups with g_B-values smaller than $\underline{j}$ in the backward side. Two issues remain to complete the identification of WVC. First, in lemmas 8–9 we will show that there exists a relation between $\underline{i}$ and $\underline{j}$. That is, if $\underline{i}$ is known, $\underline{j}$ can be seen as a function of $\underline{i}$ called $dual(\underline{i})$. Second, we will show how to find the exact pair $(\underline{i}, \underline{j})$ that will minimize WVC among all possible pairs.

Lemma 8: Let $\underline{i}$ be the minimal g_F-value for which $G_F(\underline{i}) \notin WVC$. Let $\underline{j}$ be the minimal g_B-value for which $\underline{i}+\underline{j} \geq C^*$. Then, $\underline{i}$ is also the minimal g_F-value for which $\underline{i} + \underline{j} \geq C^*$.

Proof: By negation, assume the contrary that there exists $i < \underline{i}$ such that $i + \underline{j} \geq C^*$. From the minimality of $\underline{j}$ we know that $\forall_{j<\underline{j}} i+j < \underline{i}+j < C^*$, hence $G_F(i)$ and $G_B(j)$ are a MEGP but also $G_F(\underline{i})$ and $G_B(j)$ are a MEGP. From the assumption on i we know that $\forall_{j \geq \underline{j}} i + j > i + j > i + \underline{j} \geq C^*$, hence $G_F(i)$ and $G_B(j)$ are not a MEGP and also $G_F(\underline{i})$ and $G_B(j)$ are not a MEGP. We have shown that $G_F(\underline{i})$ and $G_F(i)$ have the exact same neighbors in $\hat{G}$ ($\forall j < \underline{j}$, $G_B(j)$ is their neighbor but $\forall j \geq \underline{j}$, $G_B(j)$ is not their neighbor) so in WVC they will either both be included or both be excluded. However, from the minimality of $\underline{i}$ and from Lemma 7 we know that $G_F(\underline{i}) \notin WVC$ and $G_F(i) \in WVC$. Contradiction. □

Lemma 9: (relation between the two sides) Let $\underline{i}$ be the minimal g_F-value for which $G_F(\underline{i}) \notin WVC$. Let $\underline{j}$ be the minimal g_B-value for which $\underline{i} + \underline{j} \geq C^*$. Then, $\underline{j}$ is also the minimal g_B-value for which $G_B(\underline{j}) \notin WVC$.

Proof: From the minimality of $\underline{j}$ we know that $\forall_{j<\underline{j}}\ \underline{i}+j < C^*$ so $G_F(\underline{i})$ and $G_B(j)$ are a MEGP and since $G_F(\underline{i}) \notin$ WVC it must be that $G_B(j) \in WVC$. From Lemma 8 we know that $\underline{i}$ is the minimal g_F-value for which $\underline{i} + \underline{j} \geq C^*$. This is equivalent to the claim that $i + \underline{j} < C^*$ iff $i < \underline{i}$. Recall that $G_F(i)$ and $G_B(\underline{j})$ are a MEGP (neighbors in $\hat{G}$) iff $i + \underline{j} < C^*$ and that $G_F(i) \in WVC$ iff $i < \underline{i}$ (since $\underline{i}$ is minimal). We can conclude that all neighbors of $G_B(\underline{j})$ are included in WVC and therefore $G_B(\underline{j}) \notin WVC$. □

Algorithm 1: Calculate WVC

```
1  CalculateWVC(Ĝ, C*)
2      Run A* and find all N(G_F(i)) and N(G_B(j))
3      i = 0 j = dual(i)
4      WVC = Σ_{y<j} N(G_{B_y})
5      minWVC = WVC
6      while (i < C*) do
7          WVC = WVC + N(G_F(i))
8          i = nextg_F(i)
9          oldj = j
10         j = dual(i)
11         for (j' = j; j' < oldj; j' = nextg_B(j')) do
12             WVC = WVC − N(G_B(j'))
13         end
14         if (WVC < minWVC) then
15             minWVC = WVC
16         end
17     end
18     return (minWVC)
19 end
```

Definition 8. *For any given g_F-value i, $dual(i)$ denotes the minimal g_B-value j, for which $i + j \geq C^*$.*

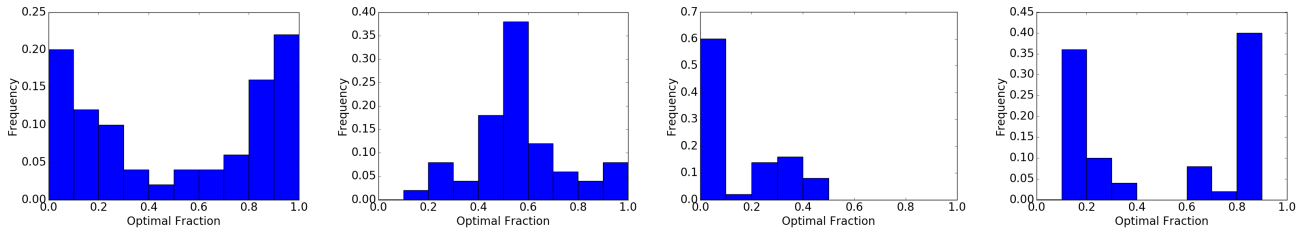
Let $\underline{i}$ be the minimal g_F-value for which $G_F(\underline{i}) \notin WVC$, and let $\underline{j}$ be the minimal g_B-value for which $G_B(\underline{j}) \notin WVC$. Lemma 9 defines their relation by proving that $\underline{j} = dual(\underline{i})$. For example, assume that in Figure 3 we have that $\underline{i} = 2$ (i.e., that $G_F(0)$ and $G_F(1)$ are in WVC but not $G_F(2)$). The minimal g_B-value j that satisfies $2 + j \geq 4$ is $j = 2$. Thus, $\underline{j} = dual(\underline{i}) = 2$ (i.e., $G_B(0)$ and $G_B(1)$ are in WVC but not $G_b(2)$).

6.5 Finding the Best Partitioning

It is important to note that C^* as well as the g_F-value $\underline{i}$ and g_B-value $\underline{j} = dual(\underline{i})$ are not known a priori. In order to find the exact minimal number of states that must be expanded one must first find $\underline{i}$ and calculate $\underline{j} = dual(\underline{i})$. The following algorithm, `CalculateWVC()`, will achieve this.

To begin, it runs forward A* and counts the number of states in each of the $G_F(i)$ groups. Next, it runs backward A* and counts the number of states in each of the $G_B(j)$ groups. Then, since A* was executed and C^* is known, $\hat{G}$

[2] Pedagogically, we may wish to define a g_F-value i such that all groups $G_F(i')$ for which $i' \leq i$ are in WVC; likewise, a g_B-value j. However, such a g-value may not exist. For example, in an A* search, only nodes from the forward side are expanded with $f < C^*$ but no nodes from the backward side are expanded all. In this case $\underline{j} = 0$, i.e., 0 is the first g-value not in WVC. Thus, we cannot talk about the maximal g_F-value in WVC.



(a) P-16(GAP-1) (b) P-16(GAP-2) (c) TOH4 (10+4 PDB) (d) Mazes

Figure 4: Distribution of p^* in different domains

can be built. Next, for each possible pair $(i, dual(i))$ accumulate the weights of the following groups:

$$Nodes(i, dual(i)) = \sum_{x<i} N(G_F(x)) + \sum_{y<j} N(G_B(y))$$

This can be done iteratively. For example, start with $nodes(0, C^*)$ and sum up all groups $G_B(j)$ for $0 \leq j < C^*$ and no group $G_F(i)$. This is equivalent to Reverse A*. Then, add the next $G_F(i)$ group but delete the corresponding $G_B(j)$ groups. This process is linear in the number of unique g-costs. The pair $(i, j = dual(i))$ whose $Nodes(i, j)$ is minimal, forms the WVC and identifies $\underline{i}$ and $\underline{j}$. The pseudocode for `CalculateWVC()` is given in Algorithm 1. The functions $nextg_F$ and $nextg_B$ give the next g_F-value and the next g_B-value. Line 7 adds the new $G_F(i)$ group while Lines 11–12 delete the corresponding $G_B(j)$ groups.

Of course, determining the exact values of $\underline{i}$ and $\underline{j} = dual(\underline{i})$ for the optimal partition can only be done post priori, after C^* is known and after $\hat{G}$ was built. But this is much simpler than building G_{MX} and storing all the individual states and edges.

7 From WVC to the Optimal Algorithm

`CalculateWVC()` calculates WVC post-priori after the groups in $\hat{G}$ are known. We now aim to find a full bidirectional search algorithm that is optimally effective.

Lemma 10: (dual is strongly restrained). Let $(i, j = dual(i))$ be a pair of g_F-value and g_B-value in $\hat{G}$. Let S_F the states from G_{MX} that belong to all $G_F(i')$ with $i' < i$ and let S_B the states from G_{MX} that belong to all $G_B(j')$ with $j' < j$. The pair of sets (S_F, S_B) is strongly restrained for $p = i/C^*$.[3]

Proof: By definition, S_F only includes states u for which $g_F(u) < i = pC^*$. In addition, S_B only includes states v for which $g_B(v) < j$. Note that $(1-p)C^* = C^* - pC^* = C^* - i$ and that from definition 8 (of $j = dual(i)$) j is the minimal g_B-value for which $1 + j \geq C^*$ so it's also the minimal g_B-value for which $j \geq C^* - i = (1-p)C^*$. We conclude that S_B only includes states v with $g_B(v) < j \geq C^* - i = (1-p)C^*$. □

Since this is true for every pair of g-values $(i, j = dual(i))$ it is also true for $(\underline{i}, \underline{j} = dual(\underline{i}))$ – the pair of g-values that define WVC. Therefore, there exists a fraction $0 \leq p^* \leq 1$ such that WVC is strongly restrained for p^*.

7.1 `fMM` that is Optimally Effective

Theorem 2. `fMM`(p^*) *is optimally effective for* $p^* = \underline{i}/C^*$

Proof: We need to show that the set of states with $f < C^*$ expanded by `fMM`(p^*) is exactly the set of states in VC. First, recall from Theorem 2 that for $0 \leq p \leq 1$, `fMM`(p) is strongly restrained for *all* nodes it expands (including those for which $f = C^*$). So, since the nodes with $f < C^*$ are a subset of these, it must be also strongly restrained for nodes with $f < C^*$. Now, in Lemma 10 we have shown that WVC is strongly restrained with $p^* = \underline{i}/C^*$. So, choosing $p^* = \underline{i}/C^*$ for `fMM` makes it strongly restrained and therefore also strongly restrained for nodes with $f < C^*$. Since choosing $p^* = \underline{i}/C^*$ gives WVC, it follows that `fMM`$(\underline{i}/C^*)$ is optimally effective. □

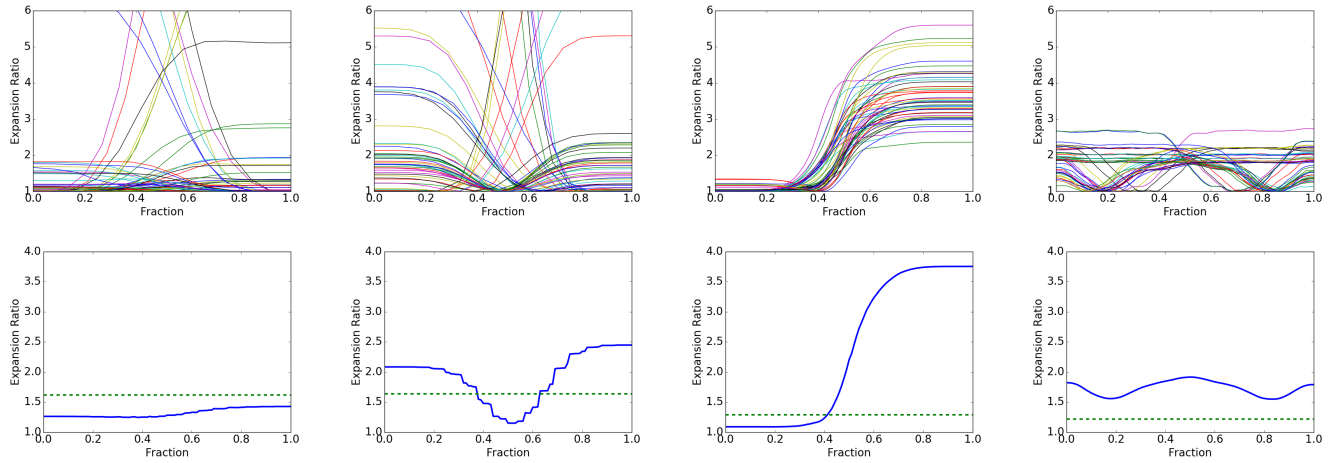
8 Experimental Results

In this section we study the impact of possible meeting points of `fMM` in a variety of domains and with a variety of heuristics. We will also compare the node expansions by various algorithms to the optimal number of nodes required. These results also provide experimental evidence around questions raised by Barker and Korf (2015) and Holte et al. (2016) as to where is the best meeting point.

We experimented on the following benchmarks with standard heuristics of different strength. **(1)** Grid-based pathfinding: 50 maze instances and instances from the 'brc' maps of Dragon Age Origins (DAO) (Sturtevant 2012). The octile heuristic is used which is relatively weak for mazes, but relatively strong on DAO maps. **(2)** 4-peg Towers of Hanoi (TOH4) problems with 14 disks (with random start states and the canonical goal state). The heuristics were additive pattern databases (Felner, Korf, and Hanan 2004) with (12+2) and (10+4) disks. **(3)** Random pancake puzzles (with the canonical goal state). The GAP heuristic (Helmert 2010) is used, which is very strong. To get a range of heuristic strengths we weakened GAP and used the GAP-x variants where the x smallest items are left out of the heuristic computation. **(4)** The standard 15 puzzle instances (Korf 1985) with the Manhattan distance heuristic.

Table 1 presents the sum of node expansions over all instances of states with $f < C^*$ divided by the size of the minimal VC of G_{MX} which was calculated using

[3] A similar lemma can be proved for j. In fact, there is a range of possible values for p. But, it is enough to show one of them.



(a) P-16(GAP-1) (b) P-16(GAP-2) (c) TOH4 (10+4 PDB) (d) Mazes

Figure 5: Ratio for individual instances (top) and for the average (bottom).

Domain	h	A*	Rev-A*	MM	NBS
P-16	GAP	1.39	**1.36**	2.11	1.92
P-16	GAP-1	1.43	**1.26**	1.39	1.62
P-16	GAP-2	2.45	2.08	**1.28**	1.63
15 puzzle	MD	1.55	**1.15**	1.44	1.43
TOH4	12+2 PDB	1.92	**1.00**	2.34	1.90
TOH4	10+4 PDB	3.75	**1.09**	2.28	1.29
Mazes	Octile	1.79	1.82	1.91	**1.22**
DAO	Octile	1.23	**1.19**	1.67	1.54

Table 1: Ratio of necessary nodes expanded and |VC|

CalculateWVC() (Algorithm 1). That is, it reports the ratio of the expansions of the algorithms vs. the minimal required. We report results for A*, Reverse A* (RevA*), MM and NBS. We note that the algorithms may continue to expand states with $f = C^*$ until a solution is found. But, as explained above, these are not regarded as must-expand states. In fact, we found that in most of these domains such extra nodes (with $f = C^*$) were never more than a few percent of the total expansions.

For each domain the best ratio is in **bold**. As expected, NBS is always less than $2 \times |\text{VC}|$, but is usually not the best. All the other algorithms had variance and for some domains they were more than $2 \times |\text{VC}|$. Thus, NBS can be seen as insurance that you will never expand more than twice the necessary nodes. RevA* outperformed regular A* in many domains because the branching factor at the fixed standard goal is the minimal possible and is smaller than the branching factor for an average state in these domains.

We then calculated the number of *must-expand* nodes for each possible p value across all instances of all domains. This corresponds to running fMM with all possible values of p. Figure 4 presents the distributions of p^* for some of these domains which illustrate the diversity of optimal meeting points. In all the frames (in this and the next figure), the x-axis corresponds to all possible meeting points (=all possible values of p). We note that the leftmost point is for the meeting point at $start$ which corresponds to RevA*. Likewise, the rightmost point is for regular A*.

With GAP-1 meeting in the middle is rarely optimal, while the exact opposite holds in GAP-2. In TOH4 RevA* dominates forward A*. On mazes, the optimal fractions follow a bimodal distribution with the peaks near 0.2 and 0.8.

Figure 5 gives the ratio of the number of must-expanded nodes for fMM(p) for each possible value of p vs. the minimal required. Individual instances are shown in the top frames and averages are in the bottom frames. The dotted line corresponds to NBS. Each individual curve must touch the bottom line at least once; this corresponds to p^* for this instance. Figure 5(a) illustrates the 50 pancake puzzle instances with the GAP-1 heuristic. Here (bottom) the exact location of p has very little effect on the average number of nodes expanded. Most instances expand relatively few nodes over the minimum, regardless of p (top). With the GAP-2 heuristic (Figure 5(b)), the best meeting point is in the middle (bottom), and it is slightly better to search backwards than forward. Most instances follow this trend (top). In the TOH4 (10+4 PDB) instances (Figure 5(c)), backwards search is best since the branching factor at the canonical goal is smaller. This is evident in both the individual instances (top) and the average performance (bottom). Figure 5(d) shows the results for 50 maze instances. Unlike the other domains, here (bottom) neither forward search, backward search or meeting in the middle is optimal. The instances (top) show the two main concentrations of optimal meeting points. It is interesting to observe that the performance NBS improves while going from left to right in Figure 5 (bottom).

Looking over all these results we see a few trends. If we know the optimal value p^*, that would always give us the best performance. But, p^* depends on the problem instances, and is not known ahead of time. What's most important is that NBS is a powerful insurance policy for situations where

the correct value of p^* is not known *a priori*.

9 Summary and Conclusions

This paper studies the *post priori* problem of minimizing node expansions in front-to-end bidirectional search, presenting an algorithm that can efficiently compute the minimum number of nodes expansions required to solve a single problem instance. This algorithm returns a fraction, p, which makes $\texttt{fMM}(p)$, a parameterized front-to-front bidirectional search algorithm, optimally effective. The bidirectional frontiers of `fMM` meet at fraction p along the optimal solution path. A*, Reverse A* and `MM` are all special cases of `fMM`. This work allows us to study the nature of bidirectional search algorithms and problem instances more deeply and allows us to measure the suboptimality of `NBS` in practice.

Future work will study ways to predict optimal values of p for different domains and problem instances using machine learning and meta-reasoning. It will also be important to study how our results change with front-to-front heuristics and how algorithms can exploit heuristics that are known to be consistent.

10 Acknowledgements

This research was supported by Israel Science Foundation (ISF) grant #417/13 and by the National Science Foundation under Grant No. 1551406.

References

Barker, J. K., and Korf, R. E. 2015. Limitations of front-to-end bidirectional heuristic search. In *Proc. 29th AAAI Conference on Artificial Intelligence*, 1086–1092.

Chen, J.; Holte, R. C.; Zilles, S.; and Sturtevant, N. R. 2017. Front-to-end bidirectional heuristic search with near-optimal node expansions. In *Proceedings of IJCAI. Also avaialble at http://arxiv.org/abs/1703.03868*.

Dechter, R., and Pearl, J. 1985. Generalized best-first search strategies and the optimality of A*. *J. ACM* 32(3):505–536.

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*.

Felner, A.; Korf, R. E.; and Hanan, S. 2004. Additive pattern database heuristics. *JAIR* 22:279–318.

Helmert, M. 2010. Landmark heuristics for the pancake problem. In *Proc. 3rd Annual Symposium on Combinatorial Search, (SoCS)*.

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 AAAI Conference on Artificial Intelligence*.

Hopcroft, J. E., and Karp, R. M. 1973. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* 2(4):225–231.

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

Sturtevant, N., and Chen, J. 2016. External memory bidirectional search. *International Joint Conference on Artificial Intelligence (IJCAI)* 676–682.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*.