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

A Theoretical Comparison of the Bounds of MM, NBS, and GBFHS

Vidal Alcázar,[1] **Mike Barley,**[2] **Pat Riddle**[2]
[1]Riken AIP, Tokyo, Japan
[2]University of Auckland, Auckland, New Zealand
vidal.alcazar@riken.jp,{barley, pat}@cs.auckland.ac.nz

Abstract

Recent work in bidirectional front-to-end heuristic search (Bi-HS) has led to the development of three different algorithms with very different behavior: MM, NBS and GBFHS. This paper presents ongoing research about their lower and upper bounds and the underlying reasons for them.

Introduction

Recently, new lines of research in Bi-HS have been explored with success. First, Holte *et al.*(2017) proposed MM[1], which uses a priority function so no nodes are expanded beyond the midpoint. MM was generalized to fMM (Shaham *et al.* 2017) so it can meet at different splits. Soon after came NBS (Chen *et al.* 2017). NBS expands at most twice the minimum number of necessarily-expanded nodes up to the last f layer, making it near-optimal. Finally, GBFHS (Barley *et al.* 2018) keeps explicit f and g limits, the latter for each side and updated by a split function. GBFHS is well-behaved (guaranteed not to expand more nodes for the same split if the heuristic improves up to the last f layer), and ensures optimality upon first collision in unit-cost domains.

Still, a clear picture of the relationship between them is missing - *i.e.* fMM and GBFHS never expand nodes beyond the meeting point, but GBFHS has other beneficial properties and a lower upper bound. Also, NBS is near-optimal, but it is unclear whether other Bi-HS algorithms may be too. We study this and, after identifying the specific conditions of these desirable properties, we propose an easy to implement and to understand algorithm that is near-optimal.

Comparing Must- and Never-Expand Nodes

In unidirectional search, any node n such that $f(n) < C^*$ must be expanded. However, in Bi-HS one side may rise C (the current lower bound on C^*) up to C^* while nodes with $f(n) < C^*$ remain unexpanded. If the heuristic is consistent and the graph is undirected, nevertheless, a set of sufficient conditions depending on C^*, ϵ and γ (the cost of the edge with the highest cost) can be found, and they are the same for both fMM and GBFHS. Lemma 1 compares MM and GBFHS with a meet-in-the-middle split function (MSF).

[1]All algorithms use ϵ if possible.

Lemma 1. *The set of must-expand nodes S_{me} of MM and GBFHS with a MSF is, $\forall n \in S_{me}$, (1) and (2) hold true:*

1. $f(n) < C^* - 2\gamma$
2. $g(n) < \dfrac{C^* - \epsilon - \gamma}{2}$

This is consistent with the experiments from Barley *et al.* (2018), in which the lower bound becomes the same as the strength of the heuristic degrades. On the other hand, in the experiments the upper bound remains consistently higher for MM. The sets of never-expand nodes are as follows:

Definition 1. *The set of never-expand nodes S_{ne} of MM is, $\forall n \in S_{ne}$, (1) or (2) holds true:*

1. $f_f(n) > C^* \wedge f_b(n) > C^*$
2. $g_f(n) > \dfrac{C^* - \epsilon}{2} \wedge g_b(n) > \dfrac{C^* - \epsilon}{2}$

Definition 2. *The set of never-expand nodes S_{ne} of GBFHS with a MSF is, $\forall n \in S_{ne}$, (1) or (2) holds true:*

1. $f_f(n) > C^* \wedge f_b(n) > C^*$
2. $g_f(n) \geq gLim_f \wedge g_b(n) \geq gLim_b$

$gLim_f = \lfloor\dfrac{C^* - \epsilon + 1}{2}\rfloor$ and $gLim_b = \lceil\dfrac{C^* - \epsilon + 1}{2}\rceil$ or *vice versa*. Hence, when $\dfrac{C^* - \epsilon}{2} = \lfloor\dfrac{C^* - \epsilon + 1}{2}\rfloor$, a whole g layer may be expanded by MM but not by GBFHS. This highlights the existence of two different g layers within the last f layer in Bi-HS, which has an important impact on the performance of the algorithm. GBFHS also commits to a whole g layer, which allows stopping upon first collision in unit-cost domains, affecting the average case too.

The meeting point of NBS is not known beforehand, but under the assumption that NBS, fMM and GBFHS meet at the same point, we can establish a comparison about S_{ne}. NBS does not impose sufficient g conditions for S_{ne}, so its S_{ne} will be equal or smaller than MM's and GBFHS's, with a worse worst-case. In fact, neither fMM nor GBFHS expand nodes beyond the meeting point, but NBS may.

Lastly, GBFHS is well-behaved because it uses g limits tied to C, which avoids committing to heuristic plateaus without raising C. This does not mean that GBFHS is superior to fMM in all cases: expanding by g means that a solution in the last f layer may be found later, so GBFHS may have to expand nodes that fMM does not.

Pruning Power of NBS

Other Bi-HS algorithms can expand nodes alternatively on both sides as NBS does. Any algorithm whose $fMin$ is monotonically increasing enforces a lower bound based on f. Thus, the advantage of NBS comes from delaying the expansion of pairs of nodes such that $g_f(u) + g_b(v) + \epsilon > C$.

Theorem 1. *Assume all expanded nodes were expanded with optimal g. Let n be a forward node and $gCMin_b = \min_{s \in Open_b} g_b(s)$ such that $f_b(s) \leq C$. If $g_f(n) + gCMin_b + \epsilon > C$ then there cannot be a solution path of cost C through n. The backwards case is analogous.*

Using Theorem 1 we propose Near-optimal Bidirectional Baseline (NBB, Algorithm 1). *Expand(d)* expands a node in direction d with minimum g among nodes n such that $f_d(n) \leq C$, updating *bestSolution*. C is updated using Theorem 1 (lines 6 and 7 of Algorithm 2). *nextFMin(C)* returns the minimum f value bigger than C in the open lists. *RunTieBreaker()* is an optional procedure invoked after increasing C that aims to quickly find a collision along paths of C cost.This allows raising C quickly while having the possibility of implementing a good tie-break for the last layer.

Algorithm 1 Near-optimal Bidirectional Baseline

```
 1: $bestSolution \Leftarrow \infty$; $C \Leftarrow 0$; $fw \Leftarrow true$
 2: while $Open_f \neq \emptyset \wedge Open_b \neq \emptyset$ do
 3:     if UpdateC() then
 4:         RunTieBreaker()
 5:     end if
 6:     if $bestSolution \leq C$ then
 7:         return $bestSolution$
 8:     end if
 9:     Expand($fw$); $fw \Leftarrow \neg fw$
10: end while
11: return $bestSolution$
```

Algorithm 2 UpdateC

```
 1: $updatedC \Leftarrow false$
 2: if $fMin > C$ then
 3:     $C \Leftarrow fMin$
 4:     $updatedC \Leftarrow true$
 5: end if
 6: while $gMin_f + gMin_b + \epsilon > C$ do
 7:     $C \Leftarrow min(gMin_f + gMin_b + \epsilon, nextFMin(C))$
 8:     $updatedC \Leftarrow true$
 9: end while
10: return $updatedC$
```

Barring *RunTieBreaker()*, NBB is near-optimal. Its behavior is analogous to NBS$_A$'s (Shperberg *et al.* 2019), so it could be seen as a version of NBS. However, NBB was developed independently from NBS$_A$ with different pseudocode and does not require priority queues for nodes, so we use *g-f* buckets (Burns *et al.* 2012) instead.

Table 1 compares NBS and NBB on: 16 Pancakes (GAP heuristic, the first k pancakes are ignored to get a weaker heuristic - 100 random problems); Korf's 15-Puzzle instances with Manhattan distance; and the Dragon Age: Origin (DAO) maps and the ten hardest mazes from Sturtevant's grid-based pathfinding benchmarks with the octile heuristic.

Domain	NBS	NBS$_n$	NBB	NBB$_n$	0-C*
Pancake (0)	**260**	68	733	**58**	4%
Pancake (1)	17162	17162	**14373**	**12624**	82%
Pancake (2)	685808	685808	**442832**	**434953**	94%
Pancake (3)	6550k	6550k	**4725k**	**4717k**	98%
15-Puzzle	12748k	12709k	**12067k**	**11739k**	86%
DAO	**11858**	11713	12092	**11661**	55%
Mazes	6558	**6556**	**6557**	**6556**	75%

Table 1: Comparison of NBS and NBB. Best result in bold.

NBS and NBS$_n$ are the average expanded and necessary ($f_x(n) < C^*$) nodes of NBS (same for NBB). 0-C* is the percentage of problems in which NBB expanded no nodes in the last f layer (NBB = NBB$_n$). As NBB tends to raise C earlier, it consistently expands similar or fewer necessary nodes than NBS. In a high number of problems NBB = NBB$_n$, so increasing C quickly leads to good performance. This makes NBB competitive in the number of expanded nodes too despite having a very bad tie-break by default, and justifies having a tie-breaking procedure for the last layer.

References

Michael W. Barley, Patricia J. Riddle, Carlos Linares López, Sean Dobson, and Ira Pohl. GBFHS: A generalized breadth-first heuristic search algorithm. In *Proceedings of the Eleventh International Symposium on Combinatorial Search, SoCS*, 2018.

Ethan Andrew Burns, Matthew Hatem, Michael J. Leighton, and Wheeler Ruml. Implementing fast heuristic search code. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search, SoCS*, 2012.

Jingwei Chen, Robert C. Holte, Sandra Zilles, and Nathan R. Sturtevant. 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.

Robert C. Holte, Ariel Felner, Guni Sharon, Nathan R. Sturtevant, and Jingwei Chen. MM: A bidirectional search algorithm that is guaranteed to meet in the middle. *Artif. Intell.*, 252, 2017.

Eshed Shaham, Ariel Felner, Jingwei Chen, and Nathan R. Sturtevant. 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.

Shahaf Shperberg, Ariel Felner, Nathan R Sturtevant, Avi Hayoun, and Eyal S Shimony. Enriching non-parametric bidirectional search algorithms. In *AAAI Conference on Artificial Intelligence*, 2019.