Bidirectional Heuristic Search in Longest Path Problems (Extended Abstract)

Tzur Shubi[1], Solomon Eyal Shimony[1], Ariel Felner[2], Shahaf S. Shperberg[2]

[1]Department of Computer Science, Ben-Gurion University of the Negev, Israel
[2]Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel
tzur.shubi@gmail.com, {shimony,felner, shperbsh}@bgu.ac.il

1 Introduction

The *longest simple path* (LSP) between two given vertices in an undirected graph is one where no vertex is visited more than once. Such problems are called Maximization (MAX) problems. LSP is known to be NP-hard, and even hard to approximate within a constant factor (Karger, Motwani, and Ramkumar 1997). A variant of LSP is *Snake-in-the-box* (SIB) (Kautz 1958), in which a path may not use neighbors of vertices that are already in the path. Treating LSP as a heuristic search problem, Stern et al. [2014] modified MIN heuristic search algorithms to solve MAX problems. Building on this idea, subsequent approaches introduced further algorithmic improvements and specialized heuristics, and extended LSP and SIB to the *generalized longest simple path* (GLSP) problem (Dahan et al. 2022).

Virtually all prior work on longest paths consists of unidirectional heuristic search algorithms (UniHS), such as A*, modified to work with MAX problems. It is well known that for MIN problems, bidirectional heuristic search (BiHS) can achieve a significant decrease in search time, especially in cases where the heuristics are moderately informative. Yet, to our knowledge, BiHS has never been applied to LSP. Adapting BiHS to the LSP problem is a challenge, requiring substantial modifications to various components, including solution detection and search bounds. Additionally, the priority functions of many BiHS algorithms become unsuitable in this context. Addressing these challenges is the focus of this paper. We present a BiHS algorithm for LSP and prove its correctness. Refinements based on the MM (meet in the middle) algorithm (Holte et al. 2017) are added, culminating in a variant called XMM, which (empirically) outperforms unidirectional search in many cases.

2 Background on GLSP and BiHS

A *state* in the state-space is a path π beginning at start vertex s. We assume, unless stated otherwise, that forward search begins with start state $\pi = (s)$, and progresses by adding one edge and vertex at a time to a path π that has s' as a last vertex. A *goal state* is a path ending at target vertex t. Following Stern et al. [2014], for a search node N we denote its state, i.e. the path it represents, by $N.\pi$, its most recently added vertex s' by $N.head$, and the rest of the vertices in $N.\pi$ by $N.tail$. For standard UniHS, $g(N)$ is the length of $N.\pi$. The successors of $N.\pi$ are denoted by $\Gamma(N.\pi)$.

In MIN problems, A* prefers nodes with **lower** $f(N) = g(N)+h(N)$ values. In GLSP, A* prefers nodes with **higher** $f(N)$ values, and h is *admissible* iff for every node N in the search space $h(N)$ is $\geq$ the weight of the longest constrained path from $N.head$ to t.

In **BiHS for MIN problems**, the objective is to find a least-cost path with cost C^* between $start$ and $goal$ in a given graph G. The shortest distance between nodes x and y is denoted by $c(x, y)$, where $c(start, goal) = C^*$. BiHS performs a forward search (F) from $start$ and a backward search (B) from $goal$, continuing until the two search frontiers meet. BiHS algorithms typically maintain two open lists, OPEN_F and OPEN_B, for the forward and backward searches, respectively. Each node is associated with $g-$, $h-$, and $f-$values. For a given direction $D \in \{F, B\}$, these values are denoted as g_D, h_D, and f_D (i.e., g_F, h_F, f_F for forward search and g_B, h_B, f_B for backward search).

Most BiHS algorithms utilize two heuristic functions, $h_F(v)$ and $h_B(v)$, which estimate $c(v, goal)$ and $c(start, v)$ for all $v \in G$, respectively. A forward heuristic h_F is said to be *admissible* if $h_F(v) \leq c(v, goal)$ for all v in G. An analogous definition applies to backward admissibility.

BiHS algorithms differ in their node- and direction-selection strategies, as well as their termination conditions. The *Meet in the Middle* (MM) algorithm (Holte et al. 2017) ensures that no node N with $g_F(N) > C^*/2$ is expanded, and uses node priorities:

$$pr_D(N) \triangleq \max(f_D(N), 2g_D(N)). \quad (1)$$

3 Bidirectional Search in GLSP

BiHS requires the ability to perform backward search from the goal state(s), as well as forward search. However, in GLSP, a state consists of an entire path. Thus, although it is easy to recognize a goal state, the number of possible goal states is exponential in the size of the input graph G, making a "standard" backward search infeasible, Instead, our backward search works by growing an initially empty path starting from the target *vertex* t rather than from a goal *state*. This makes **solution detection** non-trivial since the path constraint does not necessarily hold when a forward

path meets a backward path. **Search bounds** used to terminate search in BiHS also do not hold here.

Algorithm 1: Bidirectional LSP Algorithm (B-LSP)

```
1  OPEN_F = {MkNd(s)}, OPEN_B = {MkNd(t)}
2  U ← −1, S ← ∅
3  while (OPEN_F ∪ OPEN_B) ≠ ∅ do
4      f_max ← min_{D∈{F,B}} {max{f_D(x) | x ∈ OPEN_D}}
5      (D, N) ← argmax_{D∈{F,B}, N∈OPEN_D} f_D(N)
6      foreach M ∈ OPEN_D̄ do
7          if N.π ∩ M.π = {N.head} = {M.head}
             then
8              π ← N.π · M.π
9              if g_D(N) + g_D̄(M) > U then
10                 U ← g_D(N) + g_D̄(M)
11                 S ← π
12     if U ≥ f_max then
13         return U, S
14     remove N from OPEN_D
15     foreach π' ∈ Γ_D(N.π) do
16         add MkNd(π') to OPEN_D
17 return U, S
```

Our basic BiHS for LSP is shown as Algorithm 1. After proving the correctness of the algorithm, we add the following enhancements, **The PathMin enhancement:** analogous to *PathMAX*, but adapted to MAX problems. **Checking for a valid meeting:** done efficiently by indexing states by head-vertex and by using bit-vector representation of states.

We next attempted to use **insights from MM** in order to halt in, or near, the middle, achieved in MM by Equation 1. The analogous equation for GLSP fails, so instead we use:

$$pr_D(N) \triangleq min(2h_D(N), g_D(N) + h_D(N)) \qquad (2)$$

While delaying the expansion of nodes for which $g_D(N) > h_D(N)$, as desired, unfortunately this does not achieve the full MM guarantee of exclusively avoiding node expansions beyond the middle.

We also **cut nodes beyond the middle.** With an admissible h, f_{max} (the best f in OPEN_F and OPEN_B) is an upper bound on C^*. So no need to expand any node N with $g_D(N) \geq \frac{f_{max}}{2} \geq \frac{C^*}{2}$, because if N is on an optimal path, it will surely be met from the opposite side which will have $g_{\overline{D}}(N) \leq \frac{C^*}{2}$, where $\overline{D}$ is the side opposite to D.

Extending B-LSP to **other GLSP types** with any constraint at least as tight as "simple path" is straightforward: replace the condition in line 7 (of checking for a valid meeting) with the appropriate constraint check over the concatenated path $(N.\pi \cdot M.\pi)$. We denote the algorithm including all above enhancements by XMM.

4 Some Empirical Results

We empirically compare XMM to A* on several GLSP domains: **(1) LSP in Grids**, **(2) LSP in Mazes.**, **(3) Snake in Grids.**, and **(4) CIB (i.e. cyclic SIB).** For LSP in grids, XMM clearly outperformed unidirectional search (Fig 1), while in CIB unidirectional search was usually better (Table 1). Results were mixed in the other domains.

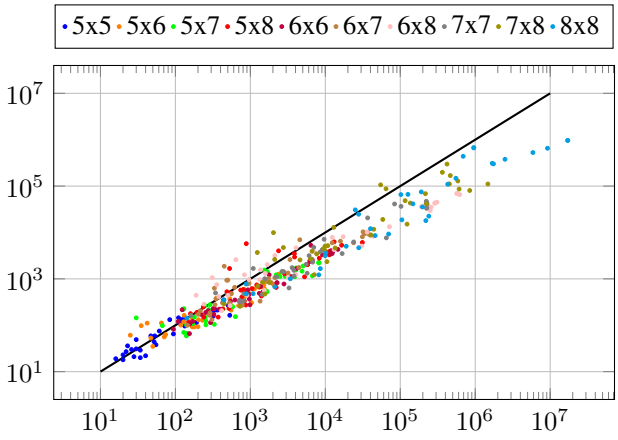

Figure 1: Expansions: XMM (Y axis) vs. A^* (X axis)

#d		Finding		Proving	
		Exp	T [ms]	Exp	T [ms]
4D	**A***	19	8	26	8
	XMM	**9**	8	**9**	8
5D	**A***	167	**141**	169	143
	XMM	**88**	142	**88**	**142**
6D	**A***	20,352	**47,266**	**20,353**	**47,398**
	XMM	**12,270**	54,114	34,231	125,033

Table 1: CIB Expansions and Runtime: XMM vs. A^*

We believe that the CIB results are due to a drastic branching factor decrease near the end of the forward search (paths near maximum length C*, resulting in fewer A^* expansions). XMM does not reach that part of the search space (path lengths only about $C^*/2$ at most), so it does not benefit from the reduced branching factor.

Acknowledgments

Supported by ISF grant #909/23.

References

Dahan, G.; Tabib, I.; Shimony, S. E.; and Felner, A. 2022. Generalized Longest Path Problems. In *SoCS*, 56–64.

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

Karger, D.; Motwani, R.; and Ramkumar, G. D. 1997. On approximating the longest path in a graph. *Algorithmica*, 18(1): 82–98.

Kautz, W. H. 1958. Unit-Distance Error-Checking Codes. *j-IRE-TRANS-ELEC-COMPUT*, EC-7(2): 179–180.

Stern, R.; Kiesel, S.; Puzis, R.; Felner, A.; and Ruml, W. 2014. Max is More than Min: Solving Maximization Problems with Heuristic Search. In *SOCS*.