AI Lab Final Report

Paper Title: *On the Completeness of Best-First Search Variants That Use Random Exploration*

Submitted By:
Amit Sarker
Roll: 99
Department of Computer Science and Engineering
University of Dhaka

Submitted To:
Dr. Md. Mosaddek Khan
Lecturer
Department of Computer Science and Engineering
University of Dhaka
&
Md. Mofijul Islam
Lecturer
Department of Computer Science and Engineering
University of Dhaka

April 22, 2019

1 Problem Description

Greedy Best First Search (GBFS) is a popular suboptimal algorithm for solving search problems. GBFS uses a greedy approach to explore a state space, which always gives priority to a node with the lowest heuristic value regardless of cost of reaching this node. Although this greedy approach can be practical, GBFS can also be misled in an arbitrary manner if heuristics are incorrect. For example, consider the Figure 1. In this graph, all nodes in the left subtree below **v** down to some depth **d** have a heuristic value of **4**. Because GBFS considers only the heuristic when evaluating nodes, the algorithm must exhaustively search the entire subtree (or heuristic plateau) before expanding **n** and finding **g**. The more misleading the heuristic (i.e. the larger the plateau), the longer it will take to solve this problem.

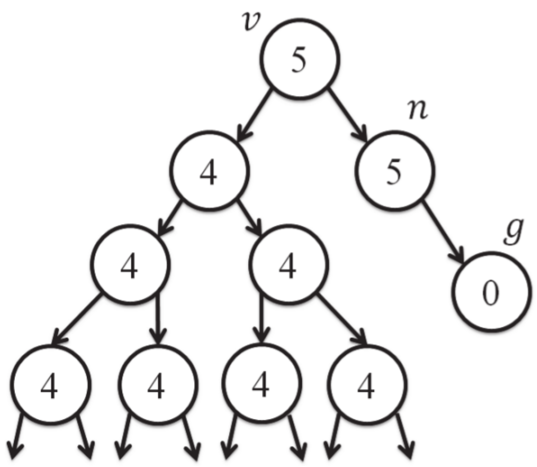

Figure 1: Graph demonstrating the weaknesses of GBFS.

2 Proposed Solution

Several approaches have enhanced GBFS through the random exploration to preserve the advantages of greediness, while strengthening the algorithm to heuristic error. **ϵ-greedy node selection** is one of them. These techniques work to encourage the search to ignore the heuristic advice from time to time. On every iteration of a GBFS that uses this technique, the algorithm will expand the node with the lowest heuristic value with probability **(1 - ϵ)**, where ϵ is a parameter such that $0 \leq \epsilon \leq 1$. Otherwise a node is randomly selected from those in the *open list* to expand. Because the resulting algorithm is still greedy for a fraction (1 - ϵ) of the time, the search will still benefit from the value of greediness and quickly pass through areas of the state-space.

3 Performance Evaluation & Results

In this paper, they did not perform any particular performance evaluation for their algorithm. They only gave the proof of the completeness of their proposed *Open-Closed List (OCL)* algorithm for finite graphs. For infinite graphs, they have proved that, the probability that a solution has been found can be made arbitrarily close to **1** given enough time. But I have done some performance evaluations of their proposed **OCL** algorithm with the standard **A* Search** algorithm and **Best-First search** algorithm based on *total number of nodes vs run time* as well as *total number of nodes vs total number of expanded nodes*. I have considered both finite and infinite (like Figure 1) graphs. Figure 2 and Figure 3 shows the comparison based on number of nodes vs run time for finite graphs. I have taken the average of 200 iterations to calculate the average run time for all the cases.

- ***Number of nodes vs Run time for finite graphs:***

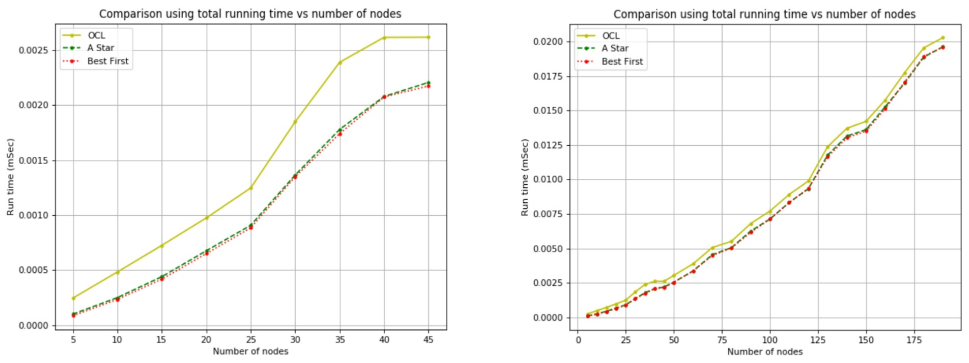

Figure 2: Number of Nodes vs Run-time for finite graphs

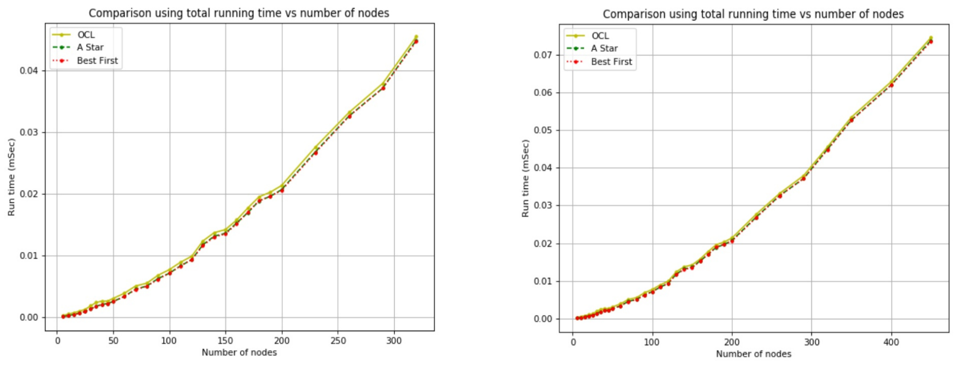

Figure 3: Number of Nodes vs Run-time for finite graphs

From Figure 2 and Figure 3, we can see that, their proposed OCL algorithm does not perform better than the standard A* and Best-First search for finite graphs when total number of nodes of the grap are small. When number of nodes are large, OCL performs like standard A* and Best-First search algorithm. The algorithm is complete as it can find the solution whenever solution is available like the standard algorithms. Which supports their claims on finite graphs. They have claimed that, on finite graphs, their proposed algorithm is complete. But they have not claimed anything about the performance of their algorithm for finite graphs.

- ***Number of nodes vs Number of expanded node during search for finite graphs:***

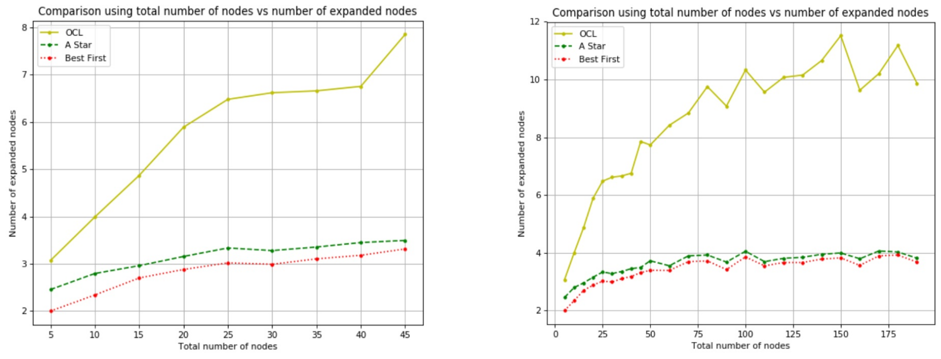

Figure 4: Number of Nodes vs Number of expanded nodes for finite graphs

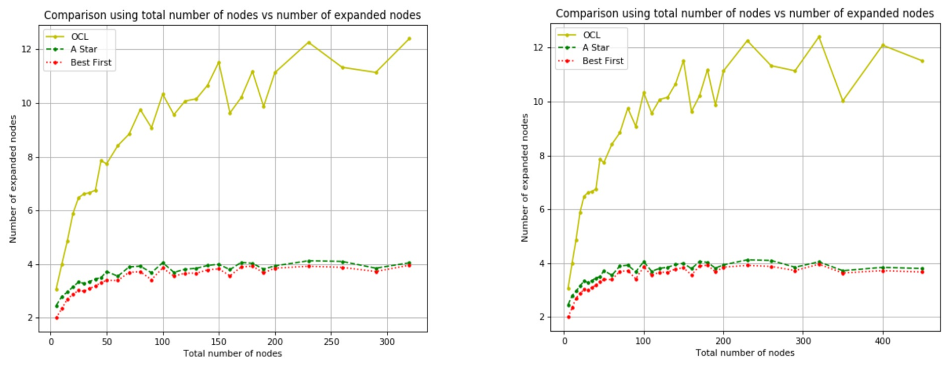

Figure 5: Number of Nodes vs Number of expanded nodes for finite graphs

Figure 4 and Figure 5 shows that, higher number of nodes are expanded during search in OCL algorithm than that of the standard algorithms for finite graph. So, for finite graphs, OCL does not perform better than the standard A* and Best-First search algorithm.

- ***Number of nodes vs Number of expanded node during search for infinite graphs:***

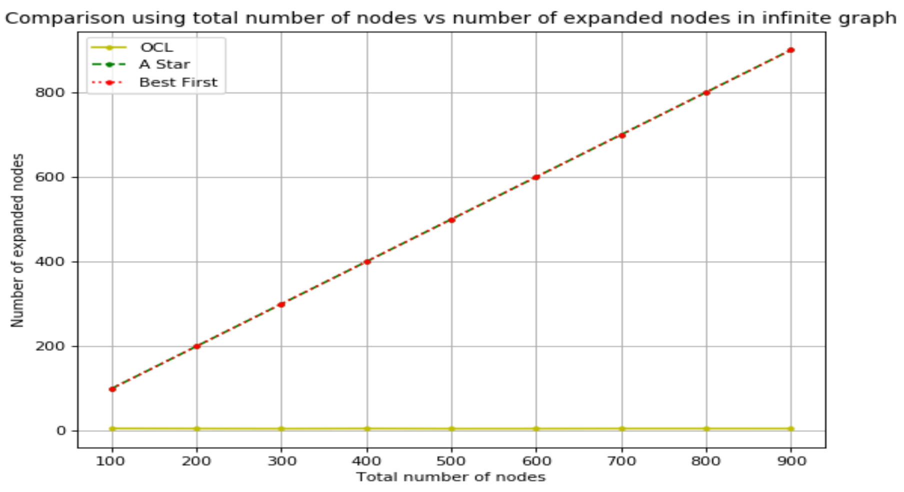

Figure 6: Number of Nodes vs Number of expanded nodes for infinite graphs

Figure 6 shows the actual strength of OCL algorithm. For infinite graphs

(Figure 1), OCL outperforms standard A* and Best-First search algorithms with a huge margin. We can see from the figure that, total number of expanded nodes are proportional to total number of nodes for standard algorithm. But for OCL, only a few number of nodes are needed to be expanded if the graph is infinite. Because for standard algorithms, if the heuristic is misleading, a large number of nodes will be expanded before reaching the goal node. But as OCL uses a stochastic method to expand the next node, only a few nodes will be expanded.

4 Summary

Table 1 and Table 2 summarizes the comparisons based on number of nodes, run time and number of expanded nodes for finite graphs. Table 3 summarizes the comparison for infinite graphs. From Table 3, we can say that, OCL outperforms the standard search algorithms for infinite graph which supports the author's claim.

Number of Nodes	Run-Time (mSec)		
	OCL	A* Search	Best-First Search
45	0.0027	0.0023	0.0022
195	0.0205	0.0195	0.0195
325	0.048	0.047	0.047
475	0.076	0.074	0.074

Table 1: Comparison of Nodes and Run-Time for finite graphs

Number of Nodes	Number of Expanded Nodes (avg)		
	OCL	A* Search	Best-First Search
45	8	4	3
195	10	4	4
325	12	4	4
475	12	4	4

Table 2: Comparison of Nodes and Expanded Nodes for finite graphs

Number of Nodes	Number of Expanded Nodes (For Infinite Graphs)		
	OCL	A* Search	Best-First Search
100	5	98	97
350	6	349	349
600	6	600	599
900	6	900	900

Table 3: Comparison of Nodes and Expanded Nodes for infinite graphs

5 Conclusion

In this paper they have considered the conduct of the best first search algorithms on infinite graphs based on several explorations. In particular, they have shown that, given enough time, it is arbitrarily likely that OCL algorithms can find a solution on the infinite graphs. This result will hold regardless of the accuracy of the heuristic information used. But for finite graph, for small number of nodes, standard algorithms are better than OCL and for large number of nodes, performance of OCL is very close to the standard algorithms.

Link of the original paper: On the Completeness of Best-First Search Variants That Use Random Exploration