Artificial Intelligence Course Work 1

Kumar Sindhurakshit , Id : 200854641

k.sindhurakshit@se21.qmul.ac.uk

School of EECS
Queen Mary University of London, UK

Program Execution : All program is included in one Jupyter note book named **agenda-based-search_ai-cw1_2011268.ipynb** and can be executed in Jupyter environment from top to bottom or with Run all cells option , Jupyter note book have details about each section . Two additional files **Analysis.csv** and **test1.csv** are included **,** Analysis.csv have detailed analysis of algorithms , while test1.csv was used for unit testing , creating various example graphs for unit testing.

Approach Summary: Significant amount of code is directly taken and derived from lab exercises with modification and additions in the code in desired functionality. Data provided in **tubedata.csv** files has been parsed with Pandas data frame and converted in Networkx Graph object for further processing by different search algorithms and provided analysis of search algorithms.

Algorithm Analysis : A csv file analysis.csv is generated using captured parameters like number of nodes visited, number of stations in path generated, execution time for each algorithms for detailed analysis. Below are few findings

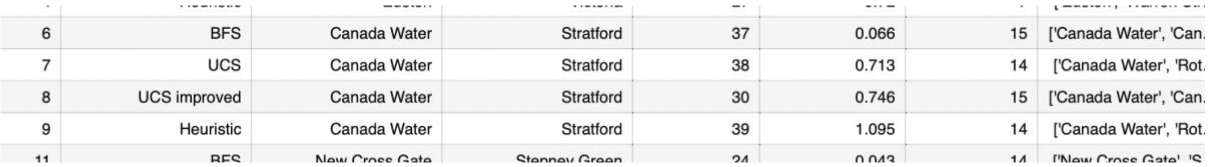
1. DFS is known to sometime get stuck in execution , and this has been also observed in my execution of recursive algorithm few source and destination combinations. I have included non-recursive with limit on visited nodes and it works fine.
2. BFS some time returned very long path , by starting traversal in unfavourable direction.
3. Extended UCS sometimes provided path with more number of stations than UCS (example Canada water to Stratford) in scenarios where UCS path returned include change of line.

Comparison DFS, BFS, and UCS

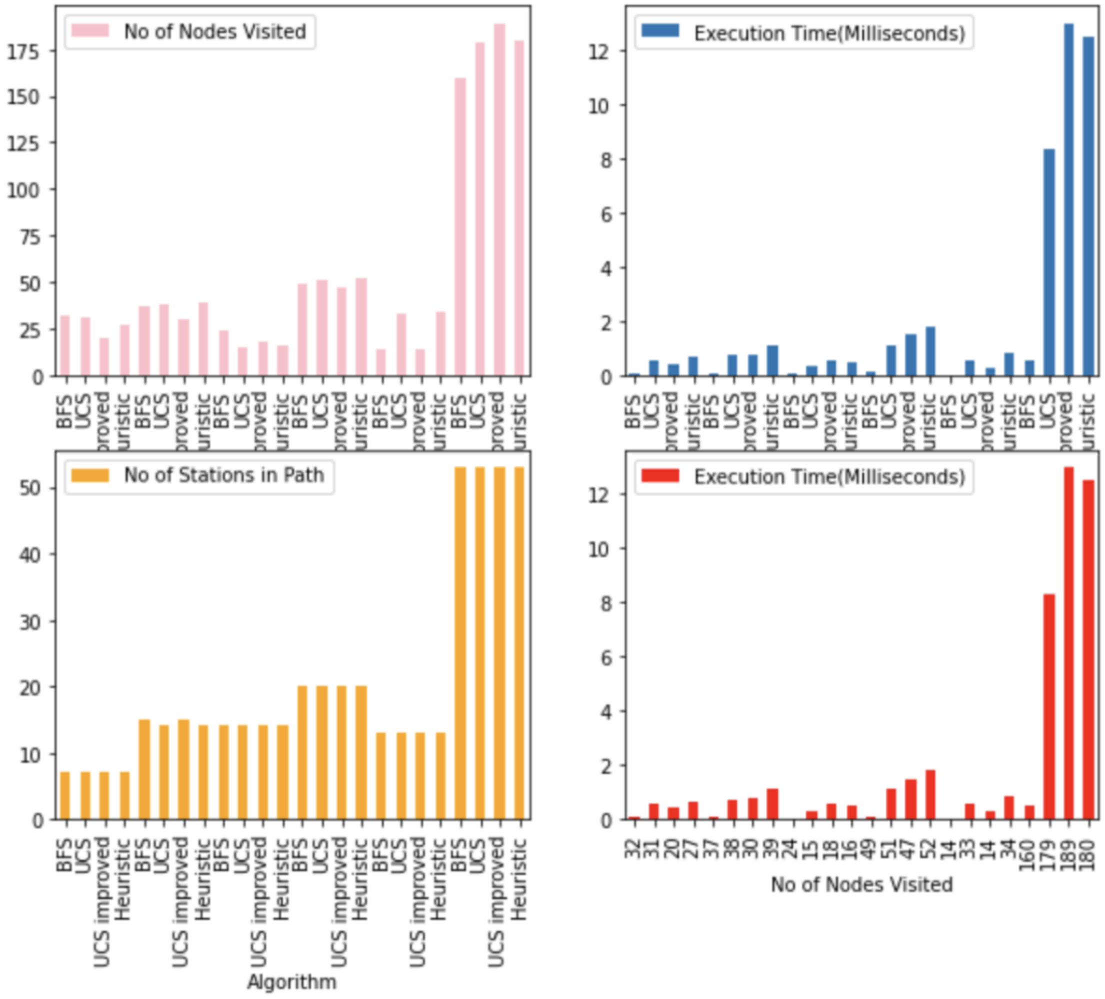
- Is any method consistently better?
    - BFS is consistently better in terms of execution time.
    - Extended UCS have more practical utility as it removed paths which require change of line.
    - Heuristics path avoid zone which are lower or higher zones then source and destination , so paths provided are optimal in terms of cost , but they are not always shortest paths.
    - DFS gets stuck too often , can we progressively change max depths ? have not implemented it though.

Cost Analysis : To cover below requirements of analysis there is a dedication section **“4 Testing and Analysis”** capturing required details and generating CSV file (analysis.csv) and graph for details analysis . All algorithms are designed to return no of visited nodes along with path.

- Returned path costs in terms of the count of the visited nodes for one route below
- Report the returned path costs in terms of the average time taken for one route below (or any route of your choice) and explain your results based on the knowledge of costs each algorithm considers.
- Report the returned path costs in terms of the visited nodes and the average time taken for one route below (or any route of your choice) for two different orders to process the nodes (direct and inverse) and explain your results for the three algorithms.
- Below is details of Canada Water to Stratford,, interesting to note that enhanced UCS returns path with 15 stations , where UCS/heuristics can do with 14 stations , it does so to avoid changing line.

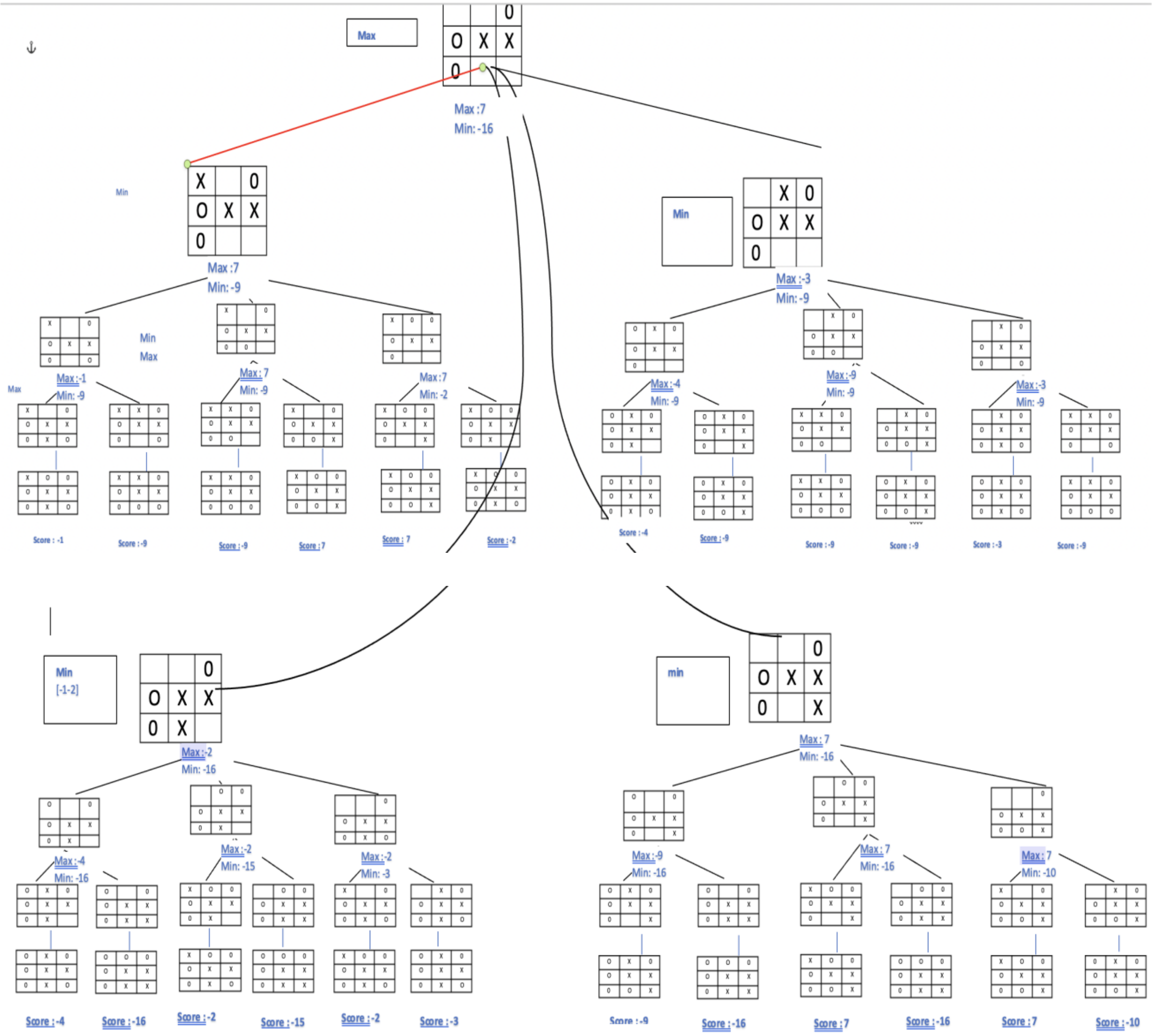


[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	BFS	Canada Water	Stratford	37	0.066	15	['Canada Water', 'Can.
7	UCS	Canada Water	Stratford	38	0.713	14	['Canada Water', 'Rot.
8	UCS improved	Canada Water	Stratford	30	0.746	15	['Canada Water', 'Can.
9	Heuristic	Canada Water	Stratford	39	1.095	14	['Canada Water', 'Rot.
11	BFS	New Cross Gate	Stepney Green	24	0.043	14	['New Cross Gate', 'S

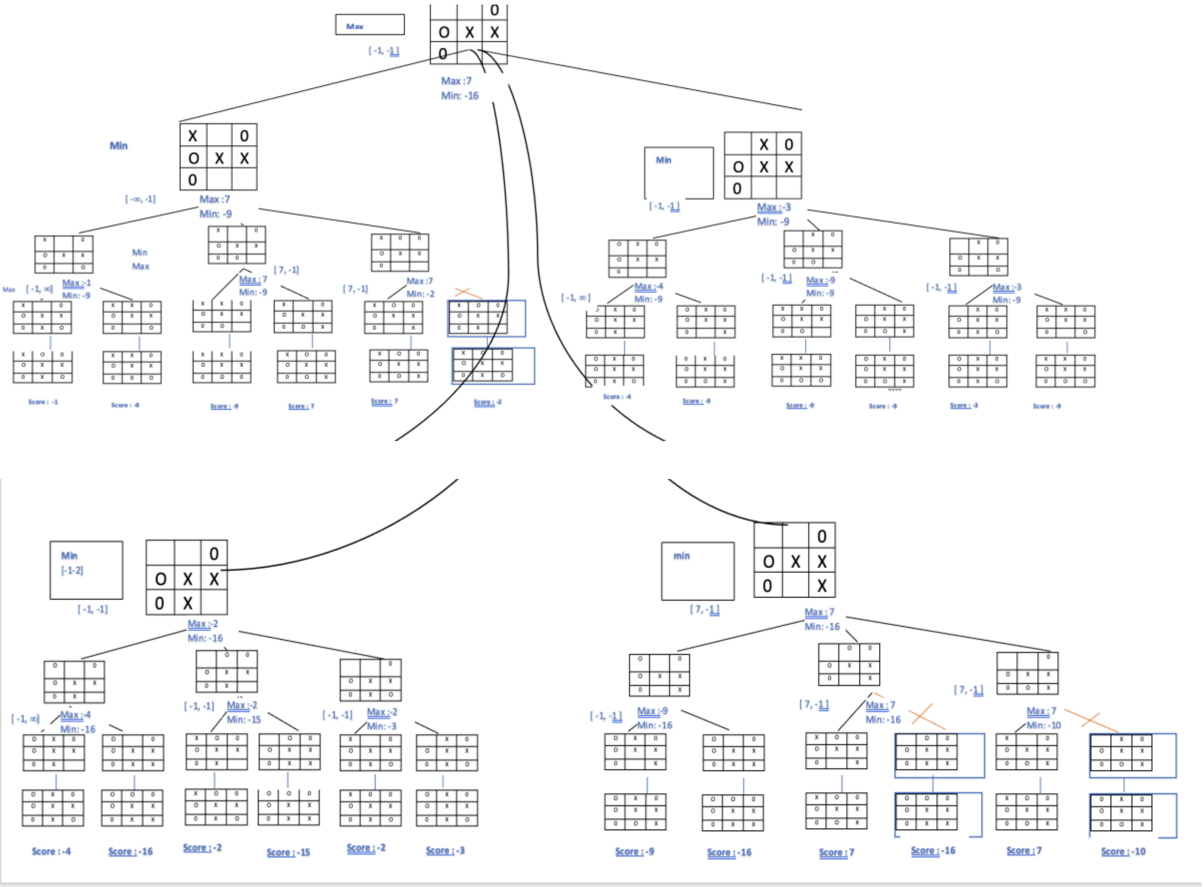


- Did face loop issues with recursive DFS , so implemented non recursive DFS with limit on maximum nodes visited , and exit if limit exceeds provided threshold. Another ideas is progressive deepening of depth to create a hybrid of BFS and DFS but have not implemented this.
- Another observation is the execution time and visited nodes grows exponentially with no of nodes visited.
-

Minimax



Alpha-Beta pruning


Max
[-1, -1]
Max :7
Min: -16
Min
[-∞, -1]
Max :7
Min: -9
Min
Max
Max
[-1, ∞]
Max :-1
Min: -9
Max : 7
Min: -9
[7, -1]
[7, -1]
Max :7
Min: -2
Score : -1
Score : -9
Score : -9
Score : 7
Score : 7
Score : -2
Min
[-1, -1]
Max :-3
Min: -9
[-1, ∞]
Max :-4
Min: -9
[-1, -1]
Max :-9
Min: -9
[-1, -1]
Max :-3
Min: -9
Score : -4
Score : -9
Score : -9
Score : -9
Score : -3
Score : -9
Min
[-1-2]
[-1, -1]
Max :-2
Min: -16
[-1, ∞]
Max :-4
Min: -16
[-1, -1]
Max :-2
Min: -15
[-1, -1]
Max :-2
Min: -3
Score : -4
Score : -16
Score : -2
Score : -15
Score : -2
Score : -3
min
[7, -1]
Max : 7
Min: -16
[-1, -1]
Max :-9
Min: -16
[7, -1]
Max : 7
Min: -16
[7, -1]
Max : 7
Min: -10
Score : -9
Score : -16
Score : 7
Score : -16
Score : 7
Score : -10