¿¿Solving 8-puzzle problem
CS7IS2 Project (2019/2020)

Abhinav Gandhi, Shikher Singh, Tanvi Bagla, Vishal Kumar

gandhia@tcd.ie, ssingh2@tcd.ie, tbagla@tcd.ie, Kumarv1@tcd.ie

Abstract. This paper approaches to solve the 8-puzzle problem, a 3x3 board with 8 tiles marked 1 to 8 and a blank square and the goal is to arrange them in order, which is a typical artificial intelligence problem. To treat this problem optimally we take four search algorithms – A Star, Uniformed Cost, Iterative Deepening Astar, and Iterative Deepening. This paper derives and tests various heuristics using the above algorithms to compare and provide the results in the form stating which algorithm can give most optimal and accurate solution to goal state. As this is a traditional problem that gives insight to several other AI problems like solving N queen puzzle, NXN puzzle and many more, solving this and empirically comparing space and time performance may give several useful insights and results that can be used in wider AI areas which motivates us to take this as an exploration.

Keywords: NEED TO CHANGE *¿¿computational geometry, graph theory, Hamilton cycles*

1 Introduction

8-Puzzle is a game invented by Sam Loyd [1] consists of 9 tiles numbered from 1 to 8 and one blank square arranged randomly. The tiles can slide horizontally or vertically in a 3X3 frame to rearrange itself in the correct order of numbers. Note that the goal state can be any fixed state that needs to be achieved. Before moving deeper let's check the problem statement.

- **Initial state-** Initial arrangement of tiles;
- **Goal state-** Final destination of the tiles;
- **Actions-** Up, Down, Left or Right w.r.t the blank square;
- **Path Cost-** Cost of each step 1. Therefore, node depth is equal to node cost.;
- **Uninformed search algorithms examined-** Breadth First Search (BFS), Uniform Cost Search (UCS), Iterative Deepening A* Search
- **Informed Search algorithms examined-** Iterative Deepening A* search

This game is a subject of continuous research where numerous algorithms can be devised to make the transition from one state to another until it reaches the goal state. Complexity is when the goal state has to be achieved in the real

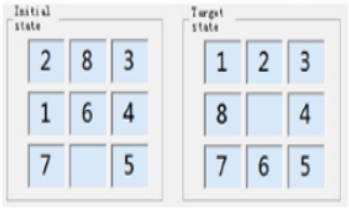

Fig. 1. ¿¿Example of a Eight-puzzle problem state chart

time providing the solution to be optimal. In this paper we study and analyze several parameters of the puzzle mentioned as follows- path_to_goal, cost_of_path, nodes_expanded, fringe_size, max_fringe_size, search_depth, max_search_depth, running_time and max_ram_usage. Depending on this search algorithms strong estimation can be made about which search algorithm performs better.

To test this problem the simplest approach is to test each move of the space. For example, in the above figure 1

This paper is organized as follows. Section II looks at related work in the areas of research. Section III outlines the problem and our approach towards the solution. Section IV details the experiment and results whereas section V presents the summary and conclusion based on results observed in section IV.

2 Related Work

There is one research where [3] N-Puzzle is solved using the distributed approach in which a problem is decomposed into sub-goals that can be treated independently, these sub goals are further divided into agents that assure those sub-goals. This method gives potential solutions and can solve large N-Puzzles. In the other work presented by Vipin Kumar, K. Ramesh and V. Nageshwara Rao Kumar [4], best first search was applied on state space graphs displaying the summary of results. Various modulations of A* Best First algorithm was implemented, stating the better performing modulations. Korf, R. E. discussed a study on Depth First search as asymptotically optimal exponential tree searches [5]. They suggest that the Depth First iterative-deepening algorithm is capable of finding an optimal solution for randomly generated 15 puzzles. Korf, R. E. presented a Linear Best First Algorithm, exploring nodes in the best first order, and expanding fewer nodes. This works on the sliding puzzle with reduced computation time, but with a penalty on solution cost.

Alexander Reinefeld mentions in his paper [2] that how IDA* is beneficial in node ordering for the 8-Puzzle problem. Authors concluded that the longest path heuristic node ordering system was most effective and fixed operator sequence worked worst.

Kuruvilla Mathew and Mujahid Tabassum [6] used Breadth First Search, Depth First Search, A* Search, Best First Search and Hill Climbing Search to

find out the optimal solution for N Puzzle game and highlighted that where Greedy BFS is memory efficient for shorter solutions, A* is suitable for longer solutions.

Noting from aforementionedabove researches, we picked a few algorithms that we wanted to compute parallel and generate comparison between them. Iterative Deepening Depth First Search was highlighted to be space and time efficient as compared to breadth first search that consumes too much space and depth first search that takes a lot more time by Korf R.E in [5] and A* and Iterative Deepening A* seemed to work best for longer solutions and randomly generated instances as mentioned in paper [2] and [6]. We took Iterative Deepening A* search, Breadth First Search (BFS), Uniform Cost Search (UCS), Iterative Deepening Depth First Search (IDDFS) as implementation for 8-Puzzle problem to test if we could find out the best out of best algorithms stated in the above researches.

3 Problem Definition and Algorithm

———**¿¿CHANGE & NEED DETAILS HERE**———— *This section formalises the problem you are addressing and the models used to solve it. This section should provide a technical discussion of the chosen/implemented algorithms. A pseudocode description of the algorithm(s) can also be beneficial to a clear explanation. It is also possible to provide one example that clarifies the way an algorithm works. It is important to highlight in this section the possible parameters involved in the model and their impact, as well as all the implementation choices that can impact the algorithm.*

3.1 Problem statement

Our problem statement is to find out the solution for 8-Puzzle problem by applying 4 search algorithms which are Breadth First Search (BFS), Uniform Cost Search (UCS), Iterative Deepening Depth First Search (IDDFS) and Iterative Deepening A* search. Our prospect is not only to solve the problem but rather to display new paradigms of solution using different search algorithms can be applied to solve efficiently instead of using traditional techniques.

3.2 Algorithms (Abhinav) keep NPuzzle

———**¿¿NEED DETAILS HERE**————
———**¿¿NEED DETAILS HERE**————
———**¿¿NEED DETAILS HERE**————

3.2.1 Uniform Cost Search (UCS)

———**¿¿NEED DETAILS HERE**————
———**¿¿NEED DETAILS HERE**————
———**¿¿NEED DETAILS HERE**————

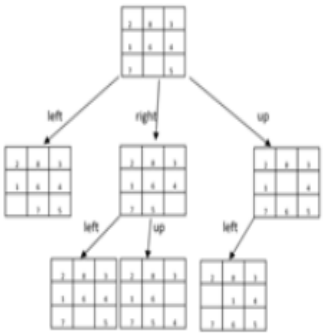

Fig. 2. ¿¿A typical approach to 8 puzzle game formed by sliding any consecutive number on blank square horizontally/vertically

3.2.2 Iterative Deepening Depth first Search (IDDFS)

¿¿A search algorithm which suffers neither the drawbacks of breadth-first nor depth-first search on trees is depth-first iterative-deepening (DFID). The algorithm works as follows: First, perform a depth-first search to depth one. Then, discarding the nodes generated in the first search, start over and do a depth-first search to level two. Next, start over again and do a depth-first search to depth three, etc., continuing this process until a goal state is reached. Since DFID expands all nodes at a given depth before expanding any nodes at a greater depth, it is guaranteed to find a shortest-length solution. Also, since at any given time it is performing a depth-first search, and never searches deeper than depth d, the space it uses is O(d).

3.2.3 A* search

The most commonly user algorithm called is A Start search. It evaluates node by combining g(n), the cost to reach the node, and h(n), the cost to get from node to the goal. F(n) = g(n) + h(n) Since g(n) giving the path cost from starting node to node n, and h(n) is the estimated cost of cheapest path from n to the goal. Hence, we are after to find the cheapest solution. That's a reason A start is considered the better because it is complete and optimal. One of the examples is shown in Figure 3.

3.3 Experimental Setup

———¿¿**NEED DETAILS HERE**———— We have performed 4 use cases and captured the results in form of below parameter for algorithms A Star, Uniformed Cost Search and Iterative Deepening Depth First Search.

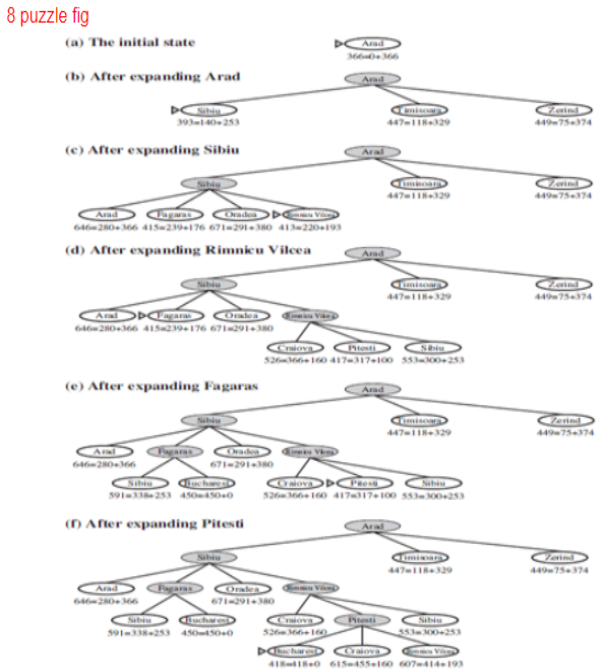

Fig. 3. ¿¿A* Search Algorithm implementation for Bucharest problem

4 Experimental Results

———¿¿**NEED DETAILS HERE**————This section should provide the details of the evaluation. Specifically:

- Methodology: describe the evaluation criteria, the data used during the evaluation, and the methodology followed to perform the evaluation.
- Results: present the results of the experimental evaluation. Graphical data and tables are two common ways to present the results. Also, a comparison with a baseline should be provided.
- Discussion: discuss the implication of the results of the proposed algorithms/models. What are the weakness/strengths of the method(s) compared with the other methods/baseline?

5 Conclusions

¿¿Provide a final discussion of the main results and conclusions of the report. Comment on the lesson learnt and possible improvements. A standard and well formatted bibliography of papers cited in the report. For example:

path_to_goal	The sequence of moves taken to reach the goal
cost_of_path	The number of moves taken to reach the goal
nodes_expanded	The number of nodes that have been expanded
search_depth	The Depth within the search tree when the goal node is found
max_search_depth	The Maximum depth of the search tree in the lifetime of the algal algorithm.
running_time	The Total running time of the search instance, reported in seconds

Fig. 4. Experimental setup

1,2,5,3,4,0,6,7,8	A STAR	UCS	IDDFS
path_to_goal	['Up','Left', 'Left']	['Up', 'Left', 'Left']	['Up', 'Left', 'Left']
cost_of_path	3	3	3
nodes_expanded	3	16	5
search_depth	3	3	3
max_search_depth	3	4	3
running_time	0.00022197	0.00037551	0.00016546
max_ram_usage	10.00000000	9.91015625	9.84375000

3,1,2,0,4,5,6,7,8	A STAR	UCS	IDDFS
path_to_goal	['Up']	['Up']	['Up']
cost_of_path	1	1	1
nodes_expanded	1	1	1
search_depth	1	1	1
max_search_depth	1	1	1
running_time	0.00011802	0.00004768	0.00003910
max_ram_usage	10.00000000	9.87890625	9.89062500

Fig. 5. Experimental result for testcases 1 and 2

References

1. Pickard S. The puzzle king: Sam Loyd's chess problems and selected mathematical puzzles. Pickard & Son Pub; 1996.
2. Reinefeld, A. 2006, Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA*, Paderborn Center for Parallel Computing, Germany.
3. Drogoul, A., and Dubreuil, C. "A distributed approach to n-puzzle solving." Proceedings of the Distributed Artificial Intelligence Workshop. 1993
4. Kumar. V. Ramesh, K. and Rao, V. N. "Parallel Best-First Search of State-Space Graphs: A Summary of Results." AAAI. Vol. 88. 1988.
5. Richard E. Korf, "Depth-First Iterative-Deepening: i z An Optimal Admissible Tree Search", Department of Computer Science, Columbia University
6. Kuruvilla Mathew and Mujahid Tabassum "Experimental Comparison of Uninformed and Heuristic AI Algorithms for N Puzzle and 8 Queen Puzzle Solution"; 2014.
7. Stuart J. Russell and Peter Norvig, Artificial Intelligence A Modern Approach Third Edition

3,1,2,4,5,0,6,7,8	A STAR	UCS	IDDFS
path_to_goal	['Left', 'Left', 'Up']	['Left', 'Left', 'Up']	['Left', 'Left', 'Up']
cost_of_path	3	3	3
nodes_expanded	3	30	11
search_depth	3	3	3
max_search_depth	3	4	3
running_time	0.00026393	0.00074506	0.00023556
max_ram_usage	10.00000000	9.86328125	9.84375000

1,2,5,0,3,4,6,7,8	A STAR	UCS	IDDFS
path_to_goal	['Right', 'Right', 'Up' eft']	['Right', 'Right', 'Up', 'Left', 'Left']	['Right', 'Right', 'Up', 'Left', 'Left']
cost_of_path	5	5	5
nodes_expanded	5	272	92
search_depth	5	5	5
max_search_depth	5	6	5
running_time	0.00036287	0.00594044	0.00223231
max_ram_usage	10.00000000	11.60156250	9.89453125

Fig. 6. Experimental result for testcase 3 and 4