Comp 6721

Artificial Intelligence

Mini Project 1 Report

Author: Student Id:

Raheleh Zarei Chamgordani 40059289

October 12, 2018

Department of Computer Science and Software Engineering

Table of Contents

1. Introduction

This report is a description for mini project one of Artificial Intelligence course at Concordia University for fall 2018 semester. Nowadays, AI has a vide variety of applications in every domain and paly a vital role in all branch of technologies. AI systems nowadays, are using in routine tasks and everywhere there is a technology we can find a footprint of AI. It has application in military, medicine, manufacturing industries, finance and economy, video games and our common home appliances.

By using AI, machine can learn how to solve a complex problem base on the algorithms that works intelligently rather than following people or animal behaviour.

1.1. Objective

Purpose of this project is to implement three main important AI algorithms, Dfs, Bfs and A* for puzzle game in a board of 3*4 tiles taking the blank tile in to account. The aim of this project is to design and implement heuristics to use with Bfs and A* algorithms and compare their efficiency as well as comparing to Dfs that is based on the depth search and do not use any heuristic. This system asks the user to enter the initial status of the unsolved 11-puzzel and would represent the final solution path in a text file by mentioning the label (each tile has been assigned by an alphabet letter from A to L) of tile that needs to be moved to reach the goal state. For the sake of this project, the moves toward the goal, has shown based on the move of non-blank tile.

1.2. Motivation

This project is a version of 8-puzzle game but in a bigger board, which is a board with 12 tiles considering the blank tiles. The board is 3X4 tiles numbered form 1 to 11. Now user is able to solve the puzzle game in a wider board and find the solution to the goal status. In this project, in addition to move the tiles in four direction, user is able to move tiles in diagonal directions as well.

2. Methodology

As the first solution in this project to solve the 11-puzzle game, the Dfs algorithm used. Base on the structure of this algorithm, a stack data structure assigned to contains open states. States in the stack wait and will be met and compared with goal status and does it for each node in open list that is not already visited until it find the solution or open list become empty. The solution path will be reflected in a text file and if algorithm was not successful to find the solution it will be return with the message "Solution not found."

For the second solution, the Bfs algorithm used but with two different heuristics. A priority queue used as the data structure to contains open states. State with height priority that is the lower heuristic value will be met at first and states with the same priority will server as first in first out. States in priority queue wait and will be visited and compared with goal status does it for each node in open list that is not already visited until it find the solution or open list become empty. The solution path [1] (amit, 2012) will be reflected in a text file and if algorithm was not successful to find the solution it will be return with the message "Solution not found."

For the third solution, the A* algorithm used but with two different heuristics that used for Bfs algorithm. Like the Bfs algorithm, a priority queue used as the data structure to contains open states. State with higher priority that is the lower heuristic value will be met at first. Each state is priority queue will be visited and compared with goal status and does it for each node in open list that is not already visited until it find the solution or open list become empty. The solution path will be reflected in a text file and if algorithm was not successful to find the solution it will be return with the message "Solution not found."

Two heuristics used for this project in Bfs and A* algorithms. First heuristic calculate the sum of the number in each tile that is out of its place in goal state. The state with lower value will be served first as a next child and compare to goal state.

3. Challenges and experiments

Challenges for this project was to design and implement admissible heuristics. A heuristic helps us to find the solution path toward the goal state more intelligently rather than by chance. The search that is done base on a heuristic is an informed search.

In designing a heuristic, we need to keep a balance between the accuracy of heuristic and how expensive it is to compute. Designing an admissible heuristic is another important thing that needs to be considered. An admissible heuristic never overestimate the cost of reaching the goal i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path.[3] (Russell, 2002)

A* algorithm must expands the lowest number of nodes than other search algorithm for a given heuristic and in this project the biggest challenge was to design heuristics to meet this requirements.

3.1 Heuristics

The first designed heuristic that calculate the sum of numbers in tiles out of place is admissible because every tile that is out of place should be moved at least once. Therefore, it creates the lowest number of nodes. (I used the idea of humming distance heuristic)

1	0	3	7
5	2	6	4
9	10	11	8

1	2	3	4
5	6	7	8
9	10	11	0

Figure 1. Tiles out of place

In this example, tiles 2, 7, 6, 4, 8 are not in their right position as the goal state so

H1 (n) = 27

In this heuristic, it is assumed that misplaces tiles can move easily from their current place in state n to their correct place in goal state. However, we can move tiles to the four sides and for diagonal direction. This is one of the disadvantage of this heuristic.

```
220     def heuristic_1(state):
221         h = 0
222         for i in range(len(state.matrix)):
223             for j in range(len((state.matrix[i]))):
224                 if state.matrix[i][j] != goalState.matrix[i][j]:
225                     h += state.matrix[i][j]
226 
227         return h
```

Figure 2.Code of heuristic one

The second designed heuristic that calculate the sum of moves that each tile out of place needs to do in order to be in its right place as goal state. It is admissible as each misplace tile moves toward its shortest path to its goal state as the diagonal move is also allowed. (The referenced article inspired me)[2] (Rok Piltaver, 2012)

Base of figure 1, the value for state n will be as follow:

H2 (n) = 2(tile 0) + 2(tile 7) + 1(tile 2) + 1(tile 6) + 1(tile 4) + 1(tile 8) = 8

```
230     def heuristic_2(state):
231         h = 0
232         for i in range(len(state.matrix)):
233             for j in range(len(state.matrix[0])):
234                 if state.matrix[i][j] != goalState.matrix[i][j]:
235                     k = i
236                     m = j
237                     index = index_2d(goalState.matrix, state.matrix[i][j])
238                     while k != index[0] and m != index[1]:
239                         if index[0] < k and index[1] < m:...
243                         elif index[0] > k and index[1] > m:...
247                         elif index[0] > k and index[1] < m:...
251                         elif index[0] < k and index[1] > m:...
255 
256                     if state.matrix[i][i] == goalState.matrix[k][m]:
257                         h
258                     else:
259                         h = h + 1
260         return h
```

Figure 3. Code for heuristic two

3.2 Comparing two heuristics

Bfs-h1			Bfs-h2			A*-h1			A*-h2		
Open Nodes	Visited Nodes	Solution path	Open Nodes	Visited Nodes	Solution path	Open Nodes	Visited Nodes	Solution path	Open Nodes	Visited Nodes	Solution path
0	1	6	0	1	6	0	1	6	0	1	6
4	2		4	2		4	2		4	2	
10	3		10	3		10	3		10	3	
16	4		16	4		16	4		16	4	
17	5		17	5		17	5		17	5	
20	6		20	6		20	6		20	6	

5. Conclusion

For the given initial path in project description (1 0 3 7 5 2 6 4 9 10 11 8) both heuristic in both Bfs and A* algorithm returned the same result. However, Dfs, by considering a depth limit of 10, find the goal state with 10 state as solution path.

References

[1] amit. (2012, October). *https://stackoverflow.com/questions*. Retrieved from https://stackoverflow.com: https://stackoverflow.com/questions/12864004/tracing-and-returning-a-path-in-depth-first-search

[2] Rok Piltaver, M. L. (2012, March). The Pathology of Heuristic Search in the 8-puzzle. *Journal of Experimental & Theoretical Artificial Intelligence*, 5.

[3] Russell, S. (2002). *Artificial Intelligence: A Modern Approach. Prentice Hall.* Norvig: ISBN 0-13-790395-2.