International Journal of Social Sciences and Scientific Studies (2023)

Content lists available on: Scholar

A Comparative Study of A* and Greedy Best-First Search Algorithms in Solving 8-Puzzle Game

Journal homepage: ijssass.com/index.php/ijssass

A Comparative Study of A* and Greedy Best-First Search Algorithms in Solving 8-Puzzle Game☆

Godswill Omonkhodion [a*]

A. Teesside Univesity, UK

Received 30 December 2022; Accepted 26 February 2023
Available online 1 March 2023

ARTICLE INFO

Keywords:

AI

Machine learning

A*

ML Algorithm

ABSTRACT

The 8-puzzle is a classic problem in the real world. It involves swapping a tile's position with an empty slot until the goal state is reached from a given initial state in the puzzle. A search algorithm helps to solve the problem of finding the shortest path between two points or two states, as in the case of this study.

In this study, we compare the performance of two well-known search algorithms, A* and greedy best-first search, in solving the 8-puzzle game. We implemented both algorithms in Python and ran experiments on test puzzle cases to evaluate their performance. The resulting outcome shows that the A* algorithm consistently outperformed the greedy best-first search algorithm in terms of both the number of moves required to reach the goal state and the running time. For most test cases, Greedy best-search could not solve the given puzzle problem within the stipulated time limit of 2 minutes.

1. Introduction

Artificial intelligence (AI) is one of the fastest-growing fields presently. AI can be defined as the ability of a machine to exhibit intelligent behaviour. Behaviour that can be comparable to or judged to exceed that of a human (Feigenbaum and Feldman, 1963). . The principal aim of AI is to develop and improve machines capable of performing tasks that would naturally require the intelligence or input of a human. Examples of these tasks are learning, problem-solving, decision-making, and language understanding. Some of the approaches to developing AI include rule-based systems and machine learning. The rule base system uses predetermined rules to guide the system's behaviour, while machine learning consists of training algorithms on data sets to enable the algorithm to make predictions or take actions (Mitchell, 1997). Machine learning techniques can be further divided into supervised learning, in which the algorithm is trained on a labelled data set, and unsupervised learning, in which the algorithm is given and data set and allowed to discover patterns that may exist in the data without explicit guidance (Biship, 2006). For example, the accuracy of disease diagnosis can be improved and made faster by the application of AI-powered diagnostic systems (Gulshan et al., 2016).

With more Self-driving cars on the road daily, road safety and traffic congestion can be improved (Shalev-Shwartz et al., 2017). However, despite the numerous positive possible applications of AI, the development and deployment of AI also raise ethical and social concerns about the potential for job displacement and bias in algorithms (Bostrom, 2014).

Many ongoing research efforts and practical applications of AI exist in various fields. For example, in natural language processing (NLP), AI systems can analyze and understand human language, enabling them to perform tasks such as translation, summarization, and text generation (Brown et al., 2020). In computer vision, AI systems can analyze and interpret images and videos, allowing them to perform object recognition and face detection (Krizhevsky et al., 2012).

There are also ongoing efforts to develop AI systems that can exhibit more general intelligence like humans. Some of such effort includes research on artificial general intelligence (AGI). This refers to the development of AI systems that can perform a wide range of tasks at a level comparable to that of humans (Legg and Hutter, 2007).

Furthermore, artificial intelligence has also found its place of application in the development of games. Many game companies are applying AI concepts in their games to improve the experience of their users. Without a doubt, AI has the potential to bring about significant advancements and improvements in many fields.

In this study, we are focused on the application of artificial intelligence search algorithms in solving an 8-puzzle game challenge. An 8-puzzle game is a sliding puzzle that consists of a 3x3 grid with eight numbered tiles and one blank space. Conventionally, the puzzle's goal is to rearrange the tiles so that they are in ascending order, with the blank space at the bottom right corner. However, in our study, we will feed the algorithm with our desired goal state and a start state. The puzzle can be solved by sliding the tiles into the blank space, with the constraint that a tile can only be moved into the blank space if it is directly above, below or beside the blank space, therefore limiting all possible movements in the game to 4 (up, down, right, and left). The 8-puzzle game has widely been used in artificial intelligence for testing and evaluating the performance of different search algorithms (Russell, S. and Norvig, P., 2003).

1.1 Search algorithms

Search algorithms are used to find a solution to a problem by exploring a search space that consists of all possible states of the problem. There are two main categories of search algorithms: uninformed search algorithms and informed search algorithms.

- Uninformed search algorithms, also known as blind search algorithms, do not use any additional information about the problem to guide the search. They rely on systematically exploring the search space to find a solution.

- On the other hand, informed search algorithms, also known as heuristic search algorithms, use additional information about the problem, such as a heuristic function, to guide the search towards a solution. In this case, the Heuristic function, denoted by h(n), determines the distance between the desired goal state and each possible state from the current state.

In this study, we compare two well-known informed search algorithms, A* and greedy best-first search, in solving the 8-puzzle game.

1.1.1 A* Search Algorithm

A* is a heuristic search algorithm that combines the strengths of both uninformed and informed search algorithms (1). It guides its search using a heuristic

function to estimate the cost of reaching the goal state from a given state and adds this estimate to the actual cost of reaching the state. The A* algorithm is an extension of the best-first search algorithm, an informed search algorithm that selects the next state to expand based on the heuristic value of the state. However, unlike the best-first search algorithm, the A* algorithm also keeps track of the actual cost of reaching the state and uses this information to guide the search. The cost of the A* algorithm is calculated with the formula

f(n) = g(n) + h(n)

where g(n) is the self-cost and h(n) is the heuristic value.

The efficiency of the A* search algorithm is based on two criteria

- Consistency: a heuristic function can be consistent if its estimated cost of reaching the goal from a particular state is equivalent to or lower than the total cost of reaching its neighbour plus the cost of reaching the goal from its neighbour.

It is mathematically represented as

$$\boldsymbol{h(n) \leq c(n,p) + h(p)}$$

where ***h(n)*** is the heuristic function cost of reaching the goal from the current state, ***n*** is a given current state, ***p*** is a possible neighbour of ***n*** and ***c(n,p)*** is the cost of reaching neighbour ***p*** from ***n***.

- Admissibility: a heuristic function can be admissible if the cost of reaching the goal is not estimated as higher than the lowest cost of reaching the goal from a given current state (Russell, S. and Norvig, P., 2003). it is mathematically represented as

$$h(n) \leq h^*(n)$$

where ***h(n)*** is the actual cost to the goal and $\boldsymbol{h^*(n)}$ is the cost of reach the goal from ***n***

1.1.2 Greedy Best-first Search Algorithm

The greedy best-first search algorithm is also an informed search algorithm like the A* algorithm. However, although it uses a heuristic function to guide the search, unlike the A* algorithm, the greedy best-first search algorithm only considers the heuristic value of the given state without considering the actual cost of reaching the state (Korf, 1985). This can lead to the algorithm getting stuck in suboptimal states and not finding the optimal solution. The cost of the greedy best-first search algorithm is calculated with the formula

f(n) = h(n)

where h(n) is the heuristic value.

2. Methodology

We implemented both A* and greedy best-first search algorithms in Python and ran experiments on test cases to evaluate their performance in solving the 8-puzzle game. The test cases were obtained from a predefined 8-puzzle problem instance online from a publication (Pillay, 2015) . For each test case, we set a time limit of two minutes (120 seconds), and we recorded the time taken by each algorithm to reach the goal (if completed or the number of moves taken if timed out), the number of moves required to reach the goal and the completeness.

We used the number of misplaced tiles as the heuristic function for both algorithms. To achieve this, we looped through the given state, comparing each tile's value with the tile's value in the same position as the goal state. For every false comparison, we initiated a counter to increase its value and this value was used as the heuristic cost (*hn*).

Index	Start State	Goal State	Minimum required steps
0	[1,2,3,8,0,4,7,6,5]	[1,3,4,8,6,2,7,0,5]	5
1	[0,3,5,4,2,8,6,1,7]	[0,1,2,3,4,5,6,7,8]	10
2	[1,2,3,8,0,4,7,6,5]	[2,8,1,4,6,3,0,7,5]	12
3	[2,3,1,7,0,8,6,5,4]	[1,2,3,8,0,4,7,6,5]	14
4	[2,3,1,8,0,4,7,6,5]	[1,2,3,8,0,4,7,6,5]	16
5	[1,2,3,8,0,4,7,6,5]	[2,3,1,8,0,4,7,6,5]	16

Table 1 shows the goal and start instances of the puzzle used for testing the algorithms.

A user interface was created using python flask package to enable the proper display of the available puzzle instances and present the returned results from the algorithm to the user.

2.1 A* Search Algorithm Pseudocode

We implemented the A* algorithm using a python set to store the already visited states and a python list for the generated children puzzles. Based on their f-values *f(n)*, which is the sum of *g(n)* (the actual cost of reaching the current state, i.e. self-cost) and *h(n)* (the heuristic estimate of the cost of reaching the goal state), a python lambda was used to sort the content of the list storing the generated children puzzles and the item at index 0 were gotten to be visited next.

The pseudocode of A* is as follows:

```
def  a_star ( startPuzzle , goalPuzzle ):
        openQueue = [startPuzzle]
        closedQueu = [ ]
        while True:
            begin
            get the current state at index 0 from openQueue and call it currentPuzzle
            if the difference in tiles between currentPuzzle and goalPuzzle is == 0:
                return the currentPuzzle as the goalPuzzle
            else:
                  begin
                   generate the possible child nodes of currentPuzzle
                   for each child node:
                       if  child node is present in closedQueu:
                           break
```

else:

begin else statement

assign the child node an f(n) value by adding the h(n) and g(n) values

add the child node to the openQueue list

end else statement

add the currentNode to the closedQueue

delete the currentNode from the openQueue

sort the openQueue based on their f(n) value

end for loop statement

end while

2.2 Greedy Best-first Search algorithm Pseudocode

Like the A* algorithm implementation, we used a python set to store the already visited states and a python list for the generated children puzzles. After that, python lambda was used to sort the content of the list storing the generated children puzzles and the item at index 0 was gotten to be visited next. However, since the greedy best-first search algorithm does not consider the self-cost, i.e. the cost of reaching the current node *g(n)*, the f-value *f(n)* was directly equivalent to *h(n)* (the heuristic estimate of the cost of reaching the goal state).

The pseudocode of Greedy Best-first is as follows:

def greedy_bfs (startPuzzle , goalPuzzle):

openQueue = [startPuzzle]

closedQueu = [ ]

while True:

begin

get the current state at index 0 from openQueue and call it currentPuzzle

if the difference in tiles between currentPuzzle and goalPuzzle is == 0:

return the currentPuzzle as the goalPuzzle

else:

begin

generate the possible child nodes of currentPuzzle

for each child node:

if child node is present in closedQueu:

break

else:

begin else statement

assign the child node an f(n) value of h(n)

add the child node to the openQueue list

end else statement

add the currentNode to the closedQueue

delete the currentNode from the openQueue

sort the openQueue based on their f(n) value

end for loop statement

end while

3. Results

Our experimental results show that the A* algorithm outperformed the greedy best-first search algorithm in most test scenarios in terms of both the number of moves required to reach the goal state and the running time, except for the first test carried out.

On average, the A* algorithm required fewer moves and had a shorter running time compared to the greedy best-first search algorithm.

For example, in one of the test cases, the A* algorithm required 38 moves to reach the goal state, while the greedy best-first search algorithm was unable to reach the goal state in the given amount of time 120 seconds (2 minutes) even after making over one thousand (1000) moves. In terms of running time, the A* algorithm took 0.02 seconds to solve the puzzle.

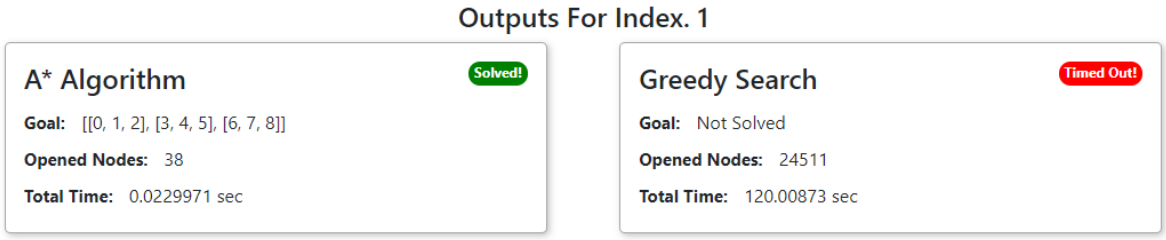

Fig 1 shows the resulting output of the index 1 test case from Table 1 above.

We ran multiple tests using the five different test cases with different difficulty levels (minimum steps), as displayed in Table 1. The table below shows the outcome of each test scenario.

Index	States	A*			Greedy BF		
		Status	Time (Seconds)	Steps	Status	Time (Seconds)	Steps
0	[1,2,3,8,0,4,7,6,5]	Solved	0.0076754	7	Solved	0.0019974	6
1	[0,3,5,4,2,8,6,1,7]	Solved	0.0419740	38	Time out	120	20502
2	[1,2,3,8,0,4,7,6,5]	Solved	0.5031149	819	Time out	120	22396
3	[2,3,1,7,0,8,6,5,4]	Solved	6.1081206	3977	Time out	120	18952
4	[2,3,1,8,0,4,7,6,5]	Time out	120	20676	Time out	120	20072
5	[1,2,3,8,0,4,7,6,5]	Time out	120	23259	Time out	120	25154

Table 2 shows the results of each test case from Table 1 above.

After the experiment was completed, it was observed that the A* algorithm could not complete the most challenging 8-puzzle instance in Table 1 (index 4 and 5) within the given amount of time.

For the first four solved puzzles, the A* algorithm had an average of 1,210 moves and a running time average of 1.6547 seconds, while greedy best search had an average of 15,465 moves and an average time of 90 seconds. These results demonstrate the superiority of the A* algorithm in solving the 8-puzzle game when compared to greedy best-first search.

Despite the close similarities in the algorithm of A* and Greedy best-first search, it is worth pointing out that one vital factor contributing to the superior performance of the A* algorithm over greedy best-first search is its ability to put into consideration its self-cost while computing f(n) value. Using the f(n) value to guide the search allows the A* algorithm to focus on states likely to lead to the goal. This helps the algorithm prevent getting stuck in suboptimal

states and allows it to find the optimal solution more efficiently.

On the other hand, the greedy best-first search algorithm's reliance on the heuristic information alone and its lack of consideration of the actual cost of reaching a given state results in the algorithm getting stuck in suboptimal states and not finding the optimal solution. In our experiments, we observed that the greedy best-first search algorithm, in most cases, often took more moves and spent more time trying to reach the goal state than the A* algorithm, indicating that it was stuck in suboptimal states.

4. Conclusion

Our study demonstrates that the A* algorithm is more effective than the greedy best-first search algorithm in solving the 8-puzzle game. The A* algorithm found the optimal solution in a shorter time and with fewer moves than the greedy best-first search algorithm. As mentioned earlier, this efficiency comes because of the mathematical formula $f(n) = h(n) + g(n)$ used by A*, which is the sum of its heuristic function and its self-cost of reaching the state. Just like in most comparison done using A*, it has yet again proven to be one of the best search algorithms.

5. Discussion

Several directions for future work can be pursued based on this study. One possibility is to compare the performance of A* and greedy best-first search on other puzzle-size games or search problems. Another direction is to use the Manhattan distance in place of the count of misplaced tiles. Manhattan distance is the sum of the vertical and horizontal distance between each tile and its position in the goal state.

6. References

(1) Russell, S. and Norvig, P. (2003). Artificial Intelligence: A Modern Approach. Prentice-Hall.

(2) Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence, 27(1), 97-109.

(3) Pillay, N. (2015) “Intelligent system design using hyper-heuristics,” *South African Computer Journal*, 56. Available at: https://doi.org/10.18489/sacj.v56i1.268.

(4) Biship, C. M. (2006). Pattern Recognition and Machine Learning (1st ed.). Springer.

(5) Bostrom, N. (2014). Superintelligence: Paths, Dangers, and Strategies. Oxford University Press.

(6) Feigenbaum, E. A., & Feldman, J. (1963). Computers and Thought. MIT Press.

(7) Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA, 316(22), 2402-

2410.

(8) Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

(9) Nguyen, A., Jo, H., & Han, S. (2015). Deep learning for healthcare: review, opportunities and challenges. Briefings in Bioinformatics, 17(2), 358-377.

(10) Shalev-Shwartz, S., Shammah, S., & Shashua, A. (2017). On the Design and Implementation of Autonomous Driving Systems. Foundations and Trends in Robotics, 6(1), 1-54.

(11) Brown, T., Mann, G. S., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Neelakantan, A. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165.

(12) Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems (pp. 1097-1105).

Legg, S., & Hutter, M. (2007). Universal Intelligence: A Definition of Machine Intelligence. Minds and Machines, 17(4), 391-444.

☆ A Comparative Study of A* and Greedy Best-First Search Algorithms in Solving 8-Puzzle Game