International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 11 – Issue 3, June 2022

Comparison of A* Algorithm and Greedy Best Search in Searching Fifteen Puzzle Solution

Charisma Tubagus Setyobudhi
Department of Computer Science, Faculty of Technology and Engineering, Diponegoro University,
Jl. Prof. Soedharto SH, Tembalang, Semarang-Indonesia
Email: charismatubagus [AT] lecturer.undip.ac.id

Abstract— **Artificial Intelligence is an exciting field to research. Artificial Intelligence itself is a broad subject. The application of artificial intelligence in daily routine is various. One of the usages of artificial intelligence is finding the shortest route on a map. In general, the algorithm which can be used for finding the shortest route is A*. A* is often used in finding the shortest route in a graph or map. Generally speaking, A* is used to make a game, especially for finding the shortest route of an intelligent agent inside it. In this paper, the finding solution of puzzle game using A* and Greedy Best First Search is to be discussed. The puzzle game which is discussed is the Fifteen Puzzle. This research compares the two algorithms used, A* and Greedy Best First Search. This research shows that Greedy Best First Search gives a slightly faster solution than A*.**

***Keywords**- A*, Artificial Intelligence, Fifteen Puzzle, Greedy Best First Search*

I. INTRODUCTION

Artificial Intelligence and Searching algorithms are needed in computer programming [1]. Searching algorithms can solve many cases. The searching algorithm can be classified into two types which are BFS(Breadth-First Search) and DFS(Depth First Search). Theoretically, those two searching algorithms are very different in finding mechanisms. Generally, these two algorithms can be used for uninformed searches to find the solution. However, some algorithms can be used if the problem has cost information. For example, in the case of finding the shortest path from one place to another place, we can use the Dijkstra algorithm and A*. A* algorithm is rapid in finding the shortest route [2].

Dijkstra Algorithm and A* are very different in calculating the cost of searching. In the Dijkstra algorithm, there is no heuristic calculation, and on the other hand, A* uses heuristic cost information. The heuristic cost value in A* is used for effectiveness and efficiency in searching solutions. A* is often used to find the shortest path in the graph or the available map. Other than finding the shortest route, A* is usually used in scheduling[3], ship path planning[4], mobile robot[5], robotic fish[6], parking guidance system[7], UAV path planning[8], road network path planning[9], autonomous land vehicle[10].

A* is a searching algorithm for the graph which contains cost information known to the user. A* has the intention to search the shortest route from one place to another place. In each looping iteration, A* has to decide which path has the least cost to explore. A* tries to minimize the searching cost which is written in the formula: $f(n) = g(n) + h(n)$ [11,12]. F(n) is the total cost that has to be considered for every node or place being explored. G(n) is the cost calculated from the origin to the current place (n). H(n) is the heuristic cost calculated from the current place to the destination. The usage of the heuristic cost function makes A* and Dijkstra different [13].

Typically, the implementation of A* is using Priority Queue for its data structure. Priority Queue is a data structure that maintains that that queue's head node is always at a minimum. A* Algorithm uses two kinds of the data structure for its storage purpose. These two data structures are the Open Set and Closed Set. Open Set can use a priority queue which has already been explained before, and open Set has the Set of nodes that the algorithm has not explored.

Meanwhile, Closed Set is the data structure in which has the Set of nodes has been explored. In general, the calculation of the heuristic of A* is various. However, the one method used frequently in calculating heuristic is the Euclidean Distance or Manhattan Distance. The pre-requisite of the heuristic function being used in calculating f(n) cost is admissible. The admissible heuristic means that the heuristic cost function never overestimates the actual cost. Some of the development of the A* algorithm is TPA*(Turning Point A*)[14], D*[15], Time-Efficient A*[16], IDA*[17]. Typically, A* can be implemented using a grid system, navigational mesh[18], or maze[19].

On the other hand, the greedy algorithm uses a similar method compared to A*. It is slightly different from A* because the greedy algorithm only uses a heuristic cost function. So the cost function can be formulated : $f(n) = h(n)$document is a template

II. RESEARCH METHOD

The application for solving Puzzles is made to find the solution to that puzzle problem. In the more extensive view of this problem, this program is made by using several steps. The steps are (1) generating the problem by reading input from text file, (2) the usage of A* and Greedy Best First Search for finding the solution, (3) the visualization of step by step solution by using OpenGL library.

To read the initial state of the problem in Fifteen Puzzle, there is a need for a text file that can be modified easily by a text editor. One instance of that text file for Puzzle's problem can be depicted below::

```
1 2 4 8
6 9 3 12
5 11 7 0
13 10 14 15
```

We can easily read the text file using the file input/output library, which C++ already provides.

Below is the pseudocode for finding a solution using the A* algorithm or Greedy Best First Search (Table 1).

Table 1. Solution Finder Pseudocode

```
void FindSolution(Pstate *initial, Pstate *goal){
  if ( !mInitStartGoal){
    clearOpenList();
    clearClosedList();
    clearPathToGoal();
  }
  SetStartAndGoal(initial, goal);
  else{
    ContinueFindPath();
  }
}
```

The searching algorithm which is being used uses iteration as the primary process. This continuously checks the state of the puzzle whether it has reached the goal state or not yet. If the goal state has not been reached, the search for a solution must continue. Both A* and Greedy Best First Search start the search process by setting the initial and goal states. This can be depicted by looking at the below pseudocode.

Table 2. Main Program Pseudocode

```
PState *initial = new PState(true);
PState *goal = new PState(false);
for (int i = 0 ; i < BOARD_W; i++)
        for(int j = 0 ; j < BOARD_H; j++)
                goal->m_CurrentState[i][j] = g_GoalState[i][j];

astar->FindSolution(initial, goal);

//Run AStar
while ( astar->m_foundGoal == false && astar->getOpenListSize() >0){
        if ( astar->iteration > MAXITERATION)
                break;
                astar->ContinueFind();
}
```

Below is the pseudocode for calculating the heuristic cost for the A* algorithm and Greedy Best First Search (Table 2).

Table 3. Heuristic Computation Pseudocode

```
void ComputeHeuristic(){
  //calculate Heuristic based on the wrong position
  H = 0;
  for( int I = 0 ; I < BOARD_W; i++)
        for ( int j = 0 ; j < BOARD_H; j++)
          if ( m_CurrentState[i][j] != g_GoalState[i][j] )
                H++;

   F = G + H;
}
```

To calculate the heuristic value, we use the computation method of using its positions that are different from the goal state. If the position of the puzzle piece is not the same as the goal state, the heuristic value will be incremented.

In implementing A* and Greedy Best First Search, there are several components whithatcome the central core of that algorithm. These components are described in the form below classes:

- PState(Puzzle State)
  PState is one component that has a function to store the puzzle state data. This PState stores the states by using a 2D array of matrices for storing the number in the puzzle pieces.
- AStar
  AStar is one component acting as the main motor engine for the A* algorithm and Greedy Best First Search. In the AStar component, several functions or methods have the task of running the algorithm.
- OGL
  OGL component is the component for drawing the PState using the OpenGL library.
- Main
  The Main class is the main component for the program in which the program is run the first time. This class is the entry point for the program to run these components can be illustrated by looking at the below diagram.

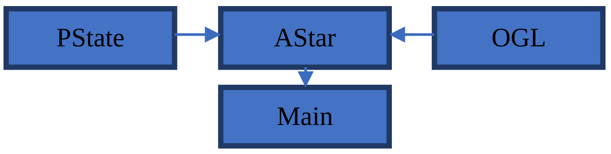

Figure 1. Program Component Classes

III. RESULT AND DISCUSSION

Below is the result of the testing of A*/Greedy Fifteen Puzzle:

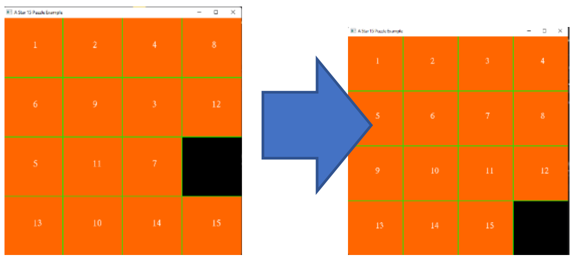

Figure 2. Initial State of Fifteen Puzzle
Figure 3. Goal State of Fifteen Puzzle

In figure 1, the initial state is the puzzle state which has been scrambled by reading the txt input file as described in the research method. Meanwhile, in figure 2, the goal state is the state of the puzzle having the number is in place (ordered) Below is the result of the searching solution step by step by using the A* algorithm(Table 4)

Table 4. A* Algorithm Step By Step Solution Result

Step	**Puzzle State**	**Cost**

1	1 2 4 8
	6 9 3 12
	5 11 7 0
	13 10 14 15
		F = 13 G = 0 H = 13

2	1 2 4 8
	6 9 3 0
	5 11 7 12
	13 10 14 15
		F = 13 G = 1 H = 12

3	1 2 4 0
	6 9 3 8
	5 11 7 12
	13 10 14 15
		F = 13 G = 2 H = 11

4	1 2 0 4
	6 9 3 8
	5 11 7 12
	13 10 14 15
		F = 13 G = 3 H = 10

5	1 2 3 4
	6 9 0 8
	5 11 7 12
	13 10 14 15
		F = 13 G = 4 H = 9

6	1 2 3 4
	6 9 7 8
	5 11 0 12
	13 10 14 15
		F = 13 G = 5 H = 8

7	1 2 3 4
	6 9 7 8
	5 0 11 12
	13 10 14 15
		F = 13 G = 6 H = 7

8	1 2 3 4
	6 0 7 8
	5 9 11 12
	13 10 14 15
		F = 14 G = 7 H = 7

9	1 2 3 4
	0 6 7 8
	5 9 11 12
	13 10 14 15
		F = 14 G = 8 H = 6

10	1 2 3 4
	5 6 7 8
	0 9 11 12
	13 10 14 15
		F = 14 G = 9 H = 5

11	1 2 3 4
	5 6 7 8
	9 0 11 12
	13 10 14 15
		F = 14 G = 10 H = 4

12	1 2 3 4
	5 6 7 8
	9 10 11 12
	13 0 14 15
		F = 14 G = 11 H = 3

13	1 2 3 4
	5 6 7 8
	9 10 11 12
	13 14 0 15
		F = 14 G = 12 H = 2

14	1 2 3 4
	5 6 7 8
	9 10 11 12
	13 14 15 0
		F = 13 G = 13 H = 0

In the second column of the above table, is the position or state of the puzzle pieces. The third colum has the calculation of the F, G and H cost. By looking at the table, we can conclude that the algorithm can find the solution which we want.
Below is the result of the searching solution step by step by using the Greedy Best First Search algorithm (Table 5)

TABLE 5. Greedy Best- Search Algorithm Step by Step Solution Result

Step	**Puzzle State**	**Cost**

1	1 2 4 8
	6 9 3 12
	5 11 7 0
	13 10 14 15
		F = 13 G = 0 H = 13

2	1 2 4 8
	6 9 3 0
	5 11 7 12
	13 10 14 15
		F = 12 G = 0 H = 12

3	1 2 4 0
	6 9 3 8
	5 11 7 12
	13 10 14 15
		F = 11 G = 0 H = 11

4	1 2 0 4
	6 9 3 8
	5 11 7 12
	13 10 14 15
		F = 10 G = 0 H = 10

5	1 2 3 4
	6 9 0 8
	5 11 7 12
	13 10 14 15
		F = 9 G = 0 H = 9

6	1 2 3 4
	6 9 7 8
	5 11 0 12
	13 10 14 15
		F = 8 G = 0 H = 8

7	1 2 3 4
	6 9 7 8
	5 0 11 12
	13 10 14 15
		F = 7 G = 0 H = 7

8	1 2 3 4
	6 0 7 8
	5 9 11 12
	13 10 14 15
		F = 7 G = 0 H = 7

9	1 2 3 4
	0 6 7 8
	5 9 11 12
	13 10 14 15
		F = 6 G = 0 H = 6

10	1 2 3 4
	5 6 7 8
	0 9 11 12
	13 10 14 15
		F = 5 G = 0 H = 5

11	1 2 3 4
	5 6 7 8
	9 0 11 12
	13 10 14 15
		F = 4 G = 0 H = 4

12	1 2 3 4
	5 6 7 8
	9 10 11 12
	13 0 14 15
		F = 3 G = 0 H = 3

13	1 2 3 4
	5 6 7 8
	9 10 11 12
	13 14 0 15
		F = 2 G = 0 H = 2

14	1 2 3 4
	5 6 7 8
	9 10 11 12
	13 14 15 0
		F = 0 G = 0 H = 0

In the second column of the above table, is the position or state of the puzzle pieces. The third colum has the calculation of the F, G and H cost. By looking at the table, we can conclude that the algorithm can find the solution which we want.
Here we have the comparison table of the number iteration being used for both A* and Greedy Best First Search algorithms (Table 6)

TABLE 6. NUMBER OF ITERATIONS USED

Algorithm Used	Number of Iterations
A*	35
Greedy Best First Search	22

CONCLUSION

From the result of this research, we can make some conclusions that both A* Algorithm and Greedy Best First Search can solve or find the solution for Fifteen Puzzle. In finding a solution, the A* algorithm has a different approach or method for finding a solution. A* uses g(n) and h(n). Meanwhile, Greedy Best First Search uses only h(n). In terms of performance, these two algorithms perform differently. Greedy Best First Search can find the solution much faster than A* because of neglecting the g(n) cost. Greedy Best First Search can outperform A* by ~37% faster (number of iterations of Greedy is 22. Meanwhile, A* is 36). This might happen because A* has search space more significant than the Greedy Best First Search because of including the g(n) cost.

REFERENCES

[1] [1]. Li xiao min, Wang Jian Ping, Ning Xin. “A* Algorithm based Robot Planning”. Applied Mechanics and Materials Vols 63-64(2011) pp 686-689.2011

[2] [2]. Kuang Ping, Luo Shuai. “A Brief Introduction of an Improved A* Search Algorithm”.2013

[3] [3]. Yang Zhong Xiu, Ren Xiao Bao, Song Jia-tao, Wang Wan-liang. “Schedule Study and Simulation Experiment. of Lift sliding Stereo Garage Based on A*”. Applied Mechanics and Materials Vol 109(2012) pp 523-527. 2011

[4] [4]. Yuanliang Zhang. "A Method of Ship's Path Planning at Sea". Applied Mechanics and Materials Vols 321-324 (2013) pp 2038-2041. 2013.

[5] [5]. Frantisek Duchon, Dominik Hunady, Martin Dekan, Andrej Babinec. "Optimal Navigation for Mobile Robot in Known Environment". Applied Mechanics and Materials Vol 282(2013) pp 33-38. 2013

[6] [6]. Huan Wang, Yulian Jiang. "Robotic Fish Path Planning based on Improved A* Algorithm". Applied Mechanics and Materials Vols 336-338(2013) pp 968-972.2013

[7] [7]. Liping Cheng, Bo Yan, Yonghai Tan. "Application of CAN bus and the Layered A* Algorithm in the Parking Guidance System". Applied Mechanics and Materials Vols 602-605 (2014) pp 887-890.2014

[8] [8]. Xia Chen, Xiangmin Chen, Jing Zhang. "The Dynamic Path Planning of UAV based on A* Algorithm". Applied Mechanics and Materials Vols 494-495 (2014) pp 1094-1097.2014

[9] [9]. Shrawan Kr Sharma, B.L. Pal. "Shortest Path Searching for Road Network Using A* Algorithm". International Journal of Computer Science and Mobile Computing Vol 4 Issue 7, July 2015, pg 513-522.2015

[10] [10]. Shang Erke, Dai Bin, Nie Yiming, Zhu Qi, Xiao Liang, Zhao Dawe. "An improved A Star based Path Planning algorithm for Autonomous Land Vehicles". International Journal of Advanced Robotic Systems September-October 2020: 1-13.2020

[11] [11]. Zhao Zhiqiang, Liu Zhihua, Hao Jiaxin. "Path Planning for Ground Simulation Object Based on A* Algorithm". Applied Mechanics and Materials Vols 229-231(2012) pp 2019-2024. 2012

[12] [12]. Xiangguang He, Yaya Wang. "Researching on AI Path Finding Algorithm in Game Development". 2012 International Symposium on Instrumentation and Measurement, Sensor Network and Automation. 2012

[13] [13]. Zhanying Zhang, Ziping Zhao. "A Multiple Mobile Robots Path Planning Algorithm Based on A* and Dijkstra".International Journal of Smart Home Vol.8 No.3 (2014) pp 75-86. 2014

[14] [14]. Han Wang, Yan Piao. "Research on Optimal Path Finding Algorithm". Applied Mechanics and Materials Vols. 427-429(2013) pp 1883-1887. 2013

[15] [15]. Dr.K. Sudhagar, M.Bala Subramanian, G.Rajarajeswari. "Path planning of Mobile Robot Agent using Heuristic Based Integrated Hybrid Algorithm". Advanced Materials Research Vols 984-985 (2014) pp 1229-1234. 2014

[16] [16]. Akshay Kumar Guruji, Himansh Agarwal, D.K. Prasediya. "Time Efficient A* Algorithm for Robot Path Planning". 3rd International Conference on Innovations in Automation and Mechatronics Engineering ICIAME 2016. 2016

[17] [17]. Anggina Primanita, Rusdi Effendi, Wahyu Hidayat. "Comparison of A* and IDA* for Non Player Character in Role Playing Game". International Conference on Electrical Engineering and Computer Science (ICECOS) 2017. 2017

[18] [18]. Yan-yan Zhang, Yan chun Shen, Li Ni Ma. "Pathfinding Algorithm of 3D Scene Based on Navigational Mesh". Advanced Materials Research Vols 1030-1032 (2014) pp 1745-1750. 2014.

[19] [19]. Milena Karova, Ivaylo Penev, Neli Kalcheva."Comparative analysis of Algorithms to search the Shortest Path in a Maze". 2016 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). 2016