2022 3rd International Conference for Emerging Technology (INCET)
Belgaum, India. May 27-29, 2022

A Comparative Analysis among Three Different Shortest Path-finding Algorithms

Anusha Aziz[1*], Sheikh Tasfia[2] and M. Akhtaruzzaman[3]
Department of Computer Science and Engineering
Military Institute of Science and Technology (MIST)
Mirpur Cantonment, Dhaka-1216, Bangladesh
[1]anusha.cse.mist@gmail.com, [2]sheikhtasfia0602@gmail.com, [3]akhter900@gmail.com

2022 3rd International Conference for Emerging Technology (INCET) | 978-1-6654-9499-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INCET54531.2022.9824074

***Abstract*—The purpose of shortest route algorithms is to find a path having fewest weights between any two vertices along the edges. However, the shortest path problem may take many different forms, requiring the employment of many algorithms. In this paper, three of the shortest path finding algorithms are implemented and a comparative analysis is conducted among the three algorithms which are A* Search, Ant Colony Optimization (ACO), and Dijkstra. The parameters considered for comparative analysis are Run-time, Diagonal Movement, Complexity, Distance, Number of turns, and Number of visited nodes that are not in the path. For all the cases A* algorithm shows the best results except for the run-time.**

***Index Terms*—Shortest Path, A* Search, ACO, Dijkstra**

I. Introduction

Shortest route algorithms seek to identify the shortest pathways among networked nodes in order to reduce routing or navigating costs. Sometimes straightforward implementations of graph theory may suggest shortest route [1] but a specific algorithm or a series of instructions in a certain sequence will allow to get the intended result [2] in an efficient way.

A computer or satellite network has N vertices (nodes or networked devices) and M edges (connecting or transmission lines) having weights which represents the transmission cost in terms of distance, delay, etc. [3]–[5]. The goal of shortest path algorithms for such network is to discover optimal path among the vertices having least amount of weights or the fewest number of hops [6]. On the other hand, for the vehicle or robot navigation, path planning is one of the most vital issues which defines finding a geometrical path from the source location of the system to its destination ensuring optimal path and no collisions [7], [8]. Mobile robots or automated vehicles are mostly used for industrial applications, transportation, military intervention, etc. [9]. In indoor environment, markers, wire grids on the floor, vision or lasers guidance are mainly used for autonomous robot navigation where shortest path algorithms play a great role [8], [10].

The A* Search algorithm is a route-finding algorithm that is used in maps (grids) to find the shortest distance between the source (starting point) and the destination (final state). It is a hybrid of low-cost-first and best-first searches strategies that incorporates path cost as well as heuristic information to decide extension of the path. A* is optimal and complete, which are the two valuable properties of search algorithms. Beside that, the Dijkstra Algorithm is a greedy uninformed searching method that is used to identify the shortest route between any two vertices (start to end). The search area extends concentrically because this method looks for the lowest-cost route among all pathways in sequence, including the starting point. Furthermore, the Ant colony optimization (ACO) technique is a probabilistic method to determine the best pathways which uses swarm intelligence and meta-heuristic strategy [11]. Swarm intelligence aims to create intelligent multi-agent systems by studying the collective behavior of social insects like ants, termites, bees, wasps, as well as other animal societies like flocks of birds or schools of fish.

However, shortest path finding algorithms are common algorithms to discuss about, but it is a demand to enrich the literature with a comparative study among uninformed, heuristic, and probabilistic methods in terms of shortest path solutions. Moreover, a few studies are identified related to this focus. Therefore, this study presents a simulated comparisons among the three algorithms based on a grid architecture.

The rest of the contents of the article are arranged as follows: Section II contains literature studies. Section III presents experimental design. Section IV explains the implementation procedures. Results and discussions are delineated in Section V. Section VI presents the limitations of the work and future recommendations. Finally, Section VII concludes the manuscript.

II. Literature Review

The A* search algorithm is successfully implemented in several research and experiments. Kusuma et al. [12] implemented A* search algorithm for humanoid robot path planning and rerouting, with satisfactory results. The algorithm also demonstrated its strength for civil Unmanned Arial Vehicle (UAV) path planning based on 3G Communication, as presented in the research conducted by Tseng et al. [13]. To solve the issue of virtual human navigation and optimal path planning, Yao et al. proposed an improved A* search algorithm using weighted cost function as $f_w(x) = \{(1-w)g(x) + wh(x)\}$ [14]. Though, a little variation is observed in the equation than the original A* algorithm, the study claimed that the modified one does not affect the admissibility of the algorithm and shows significant improvements in iteration and run-time. There are some variations of star (*) search algorithm as A*, B*, D*, Focused-D*, D*-Lite, IDA*, and SMA*, which are studied in a research conducted by Nosrati et al. [15] in terms of characteristics, methods, and approaches.

The ACO algorithm follows bio-inspired strategies based on ant's behavior. The algorithm is a meta-heuristic method which provides optimal path for swarm agents in partially known environment. Beside the shortest path-finding problems, the nurse scheduling problems was solved by using the ACO algorithm for the first time [16] through analyzing the dynamic regional problem at the Vienna Hospital compound. ACO algorithm also has exhibited some improvements over the conventional approach of graph coloring problems, as presented by Salari and Eshghi [17]. A variation of the ACO algorithm is proposed by Kashef et al. [18] as Advanced Binary ACO (ABACO) algorithm which demonstrated some improvements in their results.

The Dijkstra algorithm is a type of greedy algorithm which is commonly used in finding the shortest route of a graph, grid, or network. Wang et al. [19], used Dijkstra algorithm for a robot path planning in a maze. In another study [20], an extended version of the conventional Dijkstra algorithm was implemented to obtain a solution of optimal path-search for a car navigation system within a specified time. Kang et al. [21] also delineated a path planning algorithm using the improved Dijkstra algorithm combined with the particle swarm optimization (PSO) strategy.

III. Environment Design

A grid based environment is designed for this study, as the grid based method is commonly used for the experimentation of shortest path-finding algorithms [22]. The grid method decomposes the storage environment into a series of grid units (cells) with binary information. The proportion of the grid has a direct impact on the storage of environmental information and the complexity of the model. For this project, square grid pattern is employed to represent the navigation environment. The presented grid size is 25×25 as rows and columns having a total of 625 square cells. The horizontal direction of the grid defines the X axis, and the vertical direction of the grid defines the Y axis to establish the (X, Y) coordinate system. The white cells of the grid represent the free cells (no obstacle) and the black cells represent the cells with obstacles. START and GOAL cells are presented by Blue and Red colors, respectively. For this study, the 8-neighbors concept is considered to search the neighbours of the current cell. The algorithms are implemented using Python programming language.

IV. Implementation

In order to implement A*, ACO, and Dijkstra methods in path optimization under the storage environment, the simulation computer is configured with Intel $i5$ quad-core processor, 64-bit Win10 operating system, python 3.10.1, and pycharm community simulation software. To implement the algorithms, python's pygame library is used.

Fig1 shows the grid view where blue is the starting node and red is the ending node. All the three algorithms are implemented in the same grid environment (25×25). In Fig1, the black cells represent the blocked (obstacle) nodes and the white cells represent the free nodes.

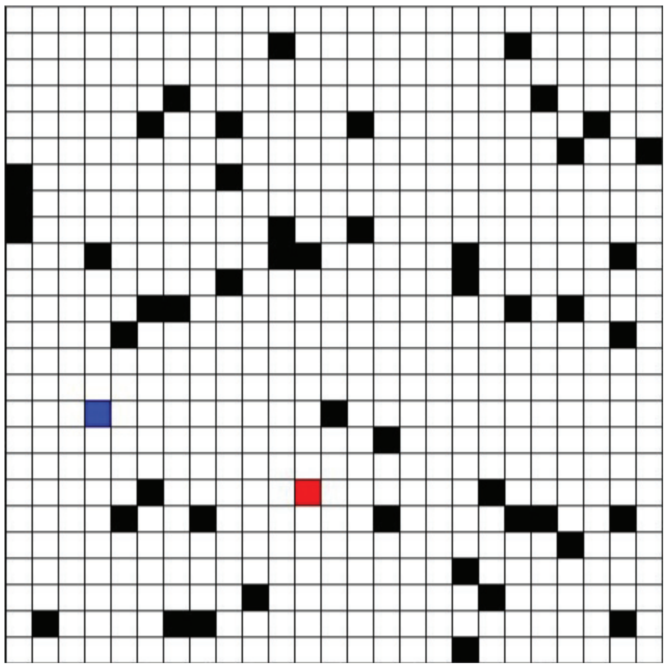
Fig. 1. Grid environment (25×25) with random placement of obstacles, White cells = no obstacle; Black cells = obstacles; Blue cell = Start position; and Red cell = Goal position

A. A* Search Algorithm

For the A* search implementation, the heuristic distance $(h(x))$ is determined by using Manhattan distance method. The steps are discussed below:

Step-1: To the list, add the start node.

Step-2: Find the F node with the lowest cost among all the eight neighbors.

Step-3: Toggle over the closed list. The eight neighbour nodes are inserted next to the present node in the open list priority queue. Ignore the node if it is not accessible.

Step-4: Continue the steps and stop working when destination is found. Destination may not be found going through all possible points.

B. ACO Algorithm

Here the total of 100 ants are considered where percentage of elite ants are 2 (2%), initial pheromone value is 1, and the amount of food is 1. The steps are presented below:

Step-1: Each ant comes up with a solution.

Step-2: The pathways discovered by several ants are compared.

Step-3: The value of a route or a pheromone is revised.

C. Dijkstra Algorithm

The steps of the Dijkstra algorithm are presented below:

Step-1: Make a zero-distance marker on the last vertex. Make this vertex as the current one.

Step-2: All vertices that lead to the present vertex should be found. Calculate the distances between them and the finish line.

Step-3: Make a note that the present vertex has been visited.

Step-4: Return to step 2 and mark the vertex with the shortest distance as current.

V. Results and Discussions

As shown in Fig2 and Fig3, the starting point (blue cell) and the end point (red cell) of the grid are artificially defined. The dark green line is the selected path on the grid. In Fig3 and Fig4, the light green grids represent the opened nodes and yellow grids represent the visited nodes. All of the three algorithms are implemented in the same grid environment.

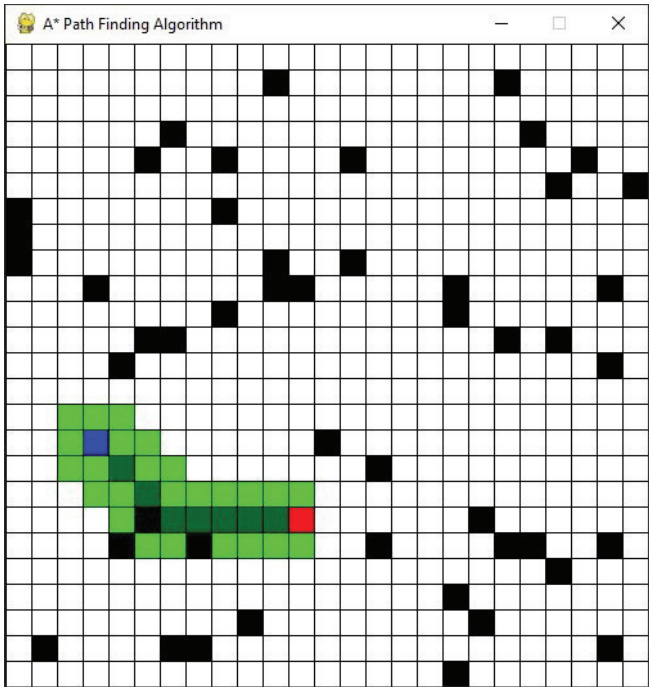

Fig. 2. Result of A* Search Algorithm

Table 1 presents the comparative results among the three algorithms. In terms of run-time, ACO Algorithm needs more time to execute while A* requires the lowest run-time among this three algorithms. As for the complexity, the Dijkstra algorithm is less complex to implement. For distance travel and number of turns, the ACO algorithm uses larger distance and grater number of turnings than the other two algorithms. In this presented experiment, the A* and Dijkstra determines same distance (shortest) paths with same number of turns. For the last parameter (number of visited nodes), ACO and A* show no extra visited cells other than the selected paths thus producing the same results. On the other hand, Dijkstra demonstrates a large number (≈ 51) of visited cells.

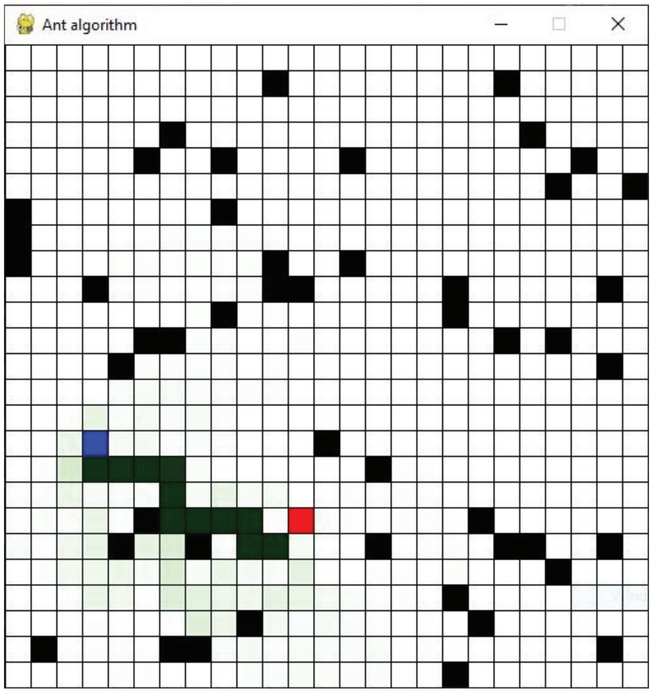

Fig. 3. Result of ACO Algorithm

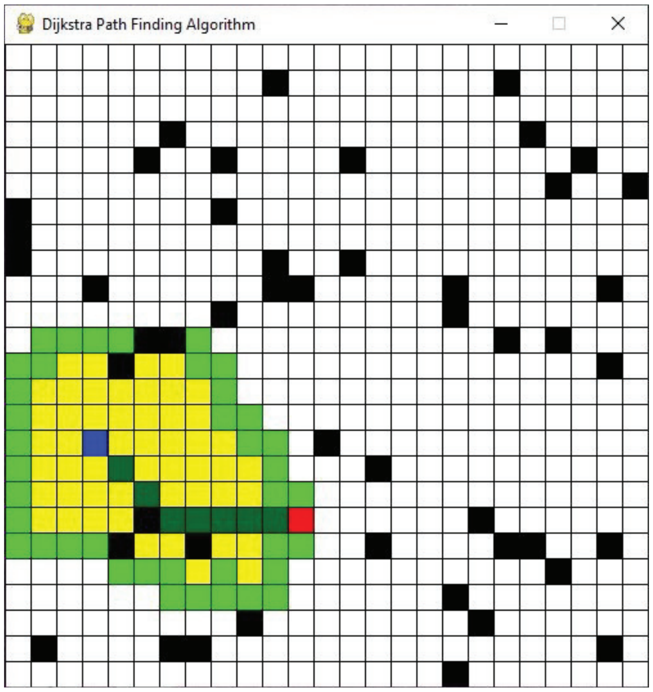

Fig. 4. Result of Dijkstra Algorithm

However, the characteristics of each of the three algorithms are different (Greedy, Heuristic, and Probabilistic), each of them has certain pros and cons in terms of the nature of the problems and scenarios. Based on the obtained results, the A* search algorithm shows better results for the demonstrated experimentation.

TABLE I
COMPARATIVE RESULTS OF THE THREE ALGORITHMS

Topic	**ACO algorithm**	**A* Search Algorithm**	**Dijkstra Algorithm**
Runtime	16 sec.	1.40 sec.	1.70 sec.
Diagonal Movement	1	4	4
Time Complexity	Comparatively more complex	$O(b^d)$	$O((V+E)logV)$
Distance	12.4 cells	9.242 cells	9.242 cells
Number of turns	7	2	2
Number of visited Nodes	0	0	51

VI. LIMITATIONS AND FUTURE RECOMMENDATIONS

The size of the grid is one of the main limitations of the study. Another limitation would be the uses of manhattan distance and euclidean distance only. Other types of distance calculation methods are not considered. In this study, only three shortest path-finding algorithms are implemented. Inclusion of several other algorithms would be a good initiative for future extension of the study.

VII. CONCLUSION

This paper presents the implementation of A*, ACO, and Dijkstra algorithms in the same grid-based (25×25) environment to conduct comparative analysis in solving the shortest path-finding problems. For the experimentation, Pycharm community software was used in the grid based environment to simulate the behavior of the algorithms. By the comparison of the obtained results, although, A* search strategy presented the best performance, it can be concluded that depending on the nature of the problems, number of agents, agent-to-agent cooperation, and environment scenario, each algorithm may behaves differently demonstrating their own pros and cons.

ACKNOWLEDGEMENTS

Authors would like to express their gratitude to the Ministry of Education, Bangladesh; Department of CSE, Military Institute of Science and Technology (MIST), Bangladesh; and DREAM Robotics Ltd., Bangladesh. The authors also would like to thank the Editors and anonymous reviewers for the insightful comments and suggestions to improve the contents of the manuscript.

REFERENCES

[1] S. E. Dreyfus, "An appraisal of some shortest-path algorithms," *Operations research*, vol. 17, no. 3, pp. 395–412, 1969.

[2] Y. N. Moschovakis, "What is an algorithm?" in *Mathematics unlimited—2001 and beyond*. Springer, 2001, pp. 919–936.

[3] B. Y. Chen, W. H. Lam, A. Sumalee, and Z.-l. Li, "Reliable shortest path finding in stochastic networks with spatial correlated link travel times," *International Journal of Geographical Information Science*, vol. 26, no. 2, pp. 365–386, 2012.

[4] F. Anwar, A. H. Abdalla, S. M. S. Bari, M. Akhtaruzzaman, M. H. Masud, J. Naeem, M. S. Azad, and M. A. Rahman, "Enhancing performance of maodv routing protocol for wireless mesh network using integrated multiple metrics technique (immt)," *International Journal of Networks and Communications*, vol. 2, pp. 1–6, 2012.

[5] M. Akhtaruzzaman, S. M. S. Bari, S. A. Hossain, and M. M. Rahman, "Link budget analysis in designing a web-application tool for military x-band satellite communication," *MIST International Journal of Science and Technology*, vol. 8, no. 1, pp. 17–33, 2020.

[6] W. Xia, C. Di, H. Guo, and S. Li, "Reinforcement learning based stochastic shortest path finding in wireless sensor networks," *IEEE Access*, vol. 7, pp. 157 807–157 817, 2019.

[7] M. Akhtaruzzaman, S. K. K. Hasan, and A. A. Shafie, "Design and development of an intelligent autonomous mobile robot for a soccer game competition," *Mechanical and Electronics Engineering*, pp. 167–171, 2009.

[8] M. Hossain, M. M. Rashid, M. M. I. Bhuiyan, S. Ahmed, and M. Akhtaruzzaman, "A qualitative approach to mobile robot navigation using rfid," *IOP Conference Series: Materials Science and Engineering*, vol. 53, pp. 1–11, 2013.

[9] M. Sanaullah, M. Akhtaruzzaman, and M. A. Hossain, "Land-robot technologies: The integration of cognitive systems in military and defense," *NDC E-JOURNAL*, vol. 2, no. 1, pp. 123–156, 2022.

[10] M. Akhtaruzzaman, A. A. Shafie, and M. Rashid, "Designing an algorithm for bioloid humanoid navigating in its indoor environment," *Journal of Mechanical Engineering and Automation*, vol. 2, no. 3, pp. 36–44, 2012.

[11] Z. Husain, A. A. Zaabi, H. Hildmann, F. Saffre, D. Ruta, and A. Isakovic, "Search and rescue in a maze-like environment with ant and dijkstra algorithms," *arXiv preprint arXiv:2111.08882*, 2021.

[12] M. Kusuma, C. Machbub *et al.*, "Humanoid robot path planning and rerouting using a-star search algorithm," in *2019 IEEE International Conference on Signals and Systems (ICSigSys)*. IEEE, 2019, pp. 110–115.

[13] F. H. Tseng, T. T. Liang, C. H. Lee, L. Der Chou, and H. C. Chao, "A star search algorithm for civil uav path planning with 3g communication," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2014, pp. 942–945.

[14] J. Yao, C. Lin, X. Xie, A. J. Wang, and C.-C. Hung, "Path planning for virtual human motion using improved a* star algorithm," in *2010 Seventh international conference on information technology: new generations*. IEEE, 2010, pp. 1154–1158.

[15] M. Nosrati, R. Karimi, and H. A. Hasanvand, "Investigation of the*(star) search algorithms: Characteristics, methods and approaches," *World Applied Programming*, vol. 2, no. 4, pp. 251–256, 2012.

[16] W. J. Gutjahr and M. S. Rauner, "An aco algorithm for a dynamic regional nurse-scheduling problem in austria," *Computers & Operations Research*, vol. 34, no. 3, pp. 642–666, 2007.

[17] E. Salari and K. Eshghi, "An aco algorithm for graph coloring problem," in *2005 ICSC Congress on computational intelligence methods and applications*. IEEE, 2005, pp. 5–pp.

[18] S. Kashef and H. Nezamabadi-pour, "An advanced aco algorithm for feature subset selection," *Neurocomputing*, vol. 147, pp. 271–279, 2015.

[19] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *2011 second international conference on mechanic automation and control engineering*. IEEE, 2011, pp. 1067–1069.

[20] M. Noto and H. Sato, "A method for the shortest path search by extended dijkstra algorithm," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3. IEEE, 2000, pp. 2316–2320.

[21] H. I. Kang, B. Lee, and K. Kim, "Path planning algorithm using the particle swarm optimization and the improved dijkstra algorithm," in *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, vol. 2. IEEE, 2008, pp. 1002–1004.

[22] J. P. Bailey, A. Nash, C. A. Tovey, and S. Koenig, "Path-length analysis for grid-based path planning," *Artificial Intelligence*, vol. 301, p. 103560, 2021.