Scientific African 15 (2022) e01068

Contents lists available at ScienceDirect

Scientific African

journal homepage: www.elsevier.com/locate/sciaf

An Improved multi-objective a-star algorithm for path planning in a large workspace: Design, Implementation, and Evaluation

Oluwaseun Opeyemi Martins, PhD[a,*], Adefemi Adeyemi Adekunle[a], Olatayo Moses Olaniyan[b], Bukola Olalekan Bolaji[c]

[a] Department of Mechatronics Engineering, Federal University, Oye-Ekiti, Nigeria
[b] Department of Computer Engineering, Federal University, Oye-Ekiti, Nigeria
[c] Department of Mechanical Engineering, Federal University, Oye-Ekiti, Nigeria

ARTICLE INFO

Article history:
Received 18 June 2021
Revised 3 November 2021
Accepted 6 December 2021

Edited by Editor Name: Dr. Oluwaseun Opeyemi Martins

Keywords:
Path planning
IMOA-star
Process time
Path length
Path smoothness

ABSTRACT

Improved path planning algorithms should minimize algorithm processing time, increase path smoothness, and shorten path length, all of which will be extremely beneficial for mobile robot traversal in large workspaces. As a result, an improved multi-objective A-star (IMOA-star) algorithm for mobile robot path planning in a large workspace was designed and implemented in Python 3.8.3 in this study. In four test cases, the proposed IMOA-star is evaluated in a large workspace with dimensions of 7120 cm × 9490 cm, and its performance is compared to the traditional A-star. When compared to the traditional A-star, the results showed that IMOA-star reduced the algorithm process time by 99.98%, improved path smoothness by 45%, reduced path length by 1.58%, and reduced the number of random points by 83.45%. Finally, the IMOA-star outperforms the traditional A-star in terms of algorithm processing time, path smoothness, path length, and the number of random points. As a result, it should be considered a viable alternative to the traditional A-star for mobile robot path planning in a large workspace.

Introduction

Path planning and localization are two components of mobile robot navigation [1]. Path planning is a process for determining an obstacle-free path for a mobile robot to travel from a start point to a goal point inside its workspace. Path planning algorithms are used to complete the work, and one of them is the A-star. Path planning algorithms calculate the best path from a starting point to a destination point based on parameters including path length, traverse time, cost, and energy consumption [2–3]. The most widely used metrics, however, are path length and traverse time [4]. Path planning algorithms for mobile robotics determine traverse points for the mobile robot in its workspace. The path of a mobile robot in 2D space is a sorted list of grid points, which is a pair of rows and columns. The first entry on the list is the path's start

* Corresponding author at: Mechatronics, Federal University Oye-Ekiti, Ikole, Nigeria.
E-mail address: oluwaseun.martins@fuoye.edu.ng (O.O. Martins).

https://doi.org/10.1016/j.sciaf.2021.e01068

point, and the last element is the path's endpoint. One of the most widely used approaches to path planning is the graph-based approach [3]. The A-star is a popular graph-based path planning method [5]. In metrics and topological configuration workspaces, the graph-based method has had a lot of success [6].

Because of their mechanical simplicity, mobile robots are favored for industrial or home applications. Nevertheless, path planning algorithms are currently being investigated. A factor in the success of such an application is finding a free path for the mobile robot to traverse through static or dynamic obstacles [7]. The A-star algorithm, like the Dijkstra and D * algorithms, is a graph-based path-finding strategy that treats the entire workspace as a grid. The A-star chooses the path that covers the fewest number of grid cells from the start point to the destination point. A-star uses the heuristic function to find the best path planning solution by computing the value of the heuristic function at each node in the workspace [8]. It also creates an obstacle map for the workspace, which includes the occupancy grid and robot radius. This technique of calculating the heuristic function at each node and constructing the obstacle map is used in every run of the A-star algorithm.

In a large workspace, however, the processing time necessary to calculate this heuristic function at each node and build the obstacle map quickly becomes prohibitive. As a result, when running A-star on a computer with a CPU configuration lower than core i5, the processing time issue is exacerbated. Another problem with the A-star algorithm is that it produces sharp turns because its heading is limited to a 45° increment, and the path it produces is not always the shortest [9]. Because non-holonomic mobile robots' kinematic constraints prevent them from following a path with sharp curves, this path with sharp twists is difficult to follow [10–11].

The remainder of this work is laid out as follows: Section II presents a review of related works. Section III describes the proposed IMOA-star algorithm. Section IV depicts the simulations and outcomes of the comparison investigation. The conclusion is found in Section V.

Related works

In recent years, many academics have been drawn to the issue of the traditional A-star algorithm's long processing time, as well as its inability to handle multi-objective problems. [3] described a method for smoothing an A-star path for easy traversal by a nonholonomic mobile robot. The presented schema smoothed the path generated by the A-star, as shown in the results, but the proposed method did not address the path search time problem. In [12], a method for determining the optimal weight of the heuristic function for the A-star algorithm was presented to minimize the path search time of the A-star algorithm by bringing the relevance of the current node's parent node to the heuristic function. The path search time was reduced by using an optimal heuristic function, but the path length was increased. In [13], an improved A-star was proposed to achieve path smoothness by including the factor of obstacle distance in the heuristic function to find a common ground between path length and path security while ignoring searching for invalid nodes that are too close to obstacles. The results show that the path smoothness was improved over the traditional A-star method, but the process time problem was not addressed. A hybrid algorithm was proposed by Zhang and Li [14]. The algorithm first employs Dijkstra's algorithm to determine the initial path, and then, in the event of a potential collision with a dynamic obstacle, the rolling window principle is used to select a local optimal target point. The A-star algorithm is used to find a new path from current location to the goal. The re-planning time was reduced when compared to traditional A-star, but there was no significant improvement in path length. Ju et al. [15] presented a guide-line-based A-star algorithm for developing the heuristic function and key points in besides obstacles to guide the A-star to avoid obstacle methods to improve traditional A-star. Although the method improved algorithm performance, no comparison with traditional A-star in terms of process time, path smoothness, and other indexes was presented. In [16], a hybrid algorithm of the A-star algorithm and the artificial potential field method was presented to avoid dynamic obstacles and find a shorter path. When the hybrid algorithm was compared to the A-star algorithm and the artificial potential field method, the proposed method reduced the path length but took longer to search the path. Liang et al. [17] presented an improved A-star algorithm designed solely for obstacle avoidance. The outcome demonstrated improved obstacle avoidance. Table 1 presents a summary of the related works discussed above.

Considering that some of the above methods address process time or other objectives such as path length and path smoothness in isolation, this paper proposes an improved multi-objective A-star (IMOA-star) method for mobile robot path planning in a large workspace. The proposed IMOA-star was implemented in Python 3.8.3. In contrast to the traditional A-star, the IMOA-star only generates the obstacle map once. The created obstacle map is saved as a pickle file and retrieved by the IMOA-star whenever a path in the workspace needs to be planned. This method has the advantage of not distorting the traditional A-star algorithm procedure. This is in contrast to previous A-star improvements in the literature, which did not determine the heuristic function value for each node [12,13]. Although it shortens the processing time of the A-star, it does so at the expense of optimal solutions. To achieve the path length and smoothness objectives, the path-problem-aware executor is included in the IMOA-star. To shorten the path, the executor evaluates the generated path and deletes random locations before forming a new line segment. It also makes sure that the angle between two line segments is as near 180° as feasible, allowing the robot to go more smoothly. Non-holonomic mobile robots will benefit from this smooth path because it is simple to follow. The IMOA-star was evaluated using four test cases, and the performance of the IMOA-star was compared to that of the A-star using processing time, path length, path smoothness, and the number of random points metrics.

Table 1
Summary of A-star related works.

Author	Title of the Work/Workdone	Approach	Limitation
[3]	Smoothed A-star Algorithm for Nonholonomic Mobile Robot Path Planning	Improved A-star algorithm for smooth path	Path search time problem was not addressed
[12]	Path Planning of Mobile Robot Based on Improved A* Algorithm	Optimal weight for the A-star heuristic function	Increased path length
[13]	Improved Safety-First A-Star Algorithm for Autonomous Vehicles	Inclusion of the factor of obstacle distance in the heuristic function	Process time problem was not addressed
[14]	Rapid path planning algorithm for mobile robot in dynamic environment	Hybrid of Dijkstra's algorithm and A-star Algorithm	No significant improvement in path length
[15]	An Improved A-Star Based Path Planning Algorithm for Autonomous Land Vehicles	A guide-line-based A-star algorithm for developing the heuristic function and key points in besides obstacles	No comparison with traditional A-star in terms of process time, path smoothness, and other indexes was presented
[16]	Path Planning Using Artificial Potential Field Method And A-star Fusion Algorithm	Hybrid of A-star algorithm and the artificial potential field method	Increased path search time
[17]	Autonomous Collision Avoidance of Unmanned Surface Vehicles Based on Improved A Star And Minimum Course Alteration Algorithms	Improved A-star algorithm for better obstacle avoidance	Path search time problem was not addressed

Material and methods

The A-star computer algorithm is a pathfinding graph search computer algorithm that is widely used [17]. The algorithm efficiently plots a feasible path on the graph between several nodes or points (source and target) [3, 16]. The A-star is an extension of Dijkstra's algorithm that employs heuristics based on knowledge of the problem at hand. However, the A-star differs from Dijkstra's because the heuristic function $h(u)$is added to the path cost function $g(u)$to define a new function;$f(u)$the estimated path cost function [14]. Using the A-star algorithm for path search for mobile robot in 2D space, the traditional A-star generates the obstacle map of the case workspace space after verifying every node in the search space. The algorithm then computes $h(u)$and displays the path list for each node in the search space that minimizes the model in Eq. (1). This results in a lengthy algorithm execution time, particularly if the workspace is large and there are many nodes to fetch. This calculation of $h(u)$and generation of an obstacle map is also repeated for each run of the traditional A-star for the same workspace and robot configuration. The proposed IMOA-star, on the other hand, addresses the difficulties observed in traditional A-star operation. It verifies each node in the search space, calculates $h(u)$and generates the obstacle map, and then saves the generated obstacle map, workspace model, and robot configuration to a database. The stored information is then used to generate the optimal path for the same robot configuration and workspace model at different source and target points at a later time. This reduces the execution time of the algorithm, allowing it to be used as a real-time path planning algorithm. Eq. (1) shows the traditional A-star model, and the flowchart below shows the procedure for the IMOA-star.

The mathematical model of the traditional A-star method is [18–19]:

$$f(u) = g(u) + h(u) \tag{1}$$

The heuristic function that represents the estimated path cost from the A-star point to a free cell is $h(u)$,$g(u)$ represents the actual route cost from a free cell to the goal point. $f(u)$is the estimated path cost from the A-star point through free cells to the target point. Manhattan, diagonal, or Euclidean distance are all terms used to describe the heuristic function [20].

The proposed IMOA-star depicted in Fig. 1 is implemented in Python 3.8.3. The IMOA-star generates the obstacle map once as a Python object structure. The Python object structure is converted to a byte stream and saved as a pickle file. The pickle file's storage format is the filename pickle_rr_robot radius. The filename is the name of the workspace 2D picture made using Inkscape-0.92.1, and the robot radius is measured in centimeters. After preparing and saving the pickle file for a workspace and the robot radius, planning a path on the IMOA-star for the same or a different start point and goal point using the same workspace model and robot radius eliminates the need to create the obstacle map again. The IMOA-star downloads the pickle file and de-serializes the previously serialized byte stream to a Python object structure, generating a path list along the way. However, before IMOA-star prints the path list, the path is examined by the included path-problem-aware executor to reduce path length and increase path smoothness.

Path length objective

The goal is to create a path that is as short as possible, hence the path length is a goal to be minimized. The evaluated euclidean distance between the extreme points of a segment is its length. Where the euclidean distance between two points

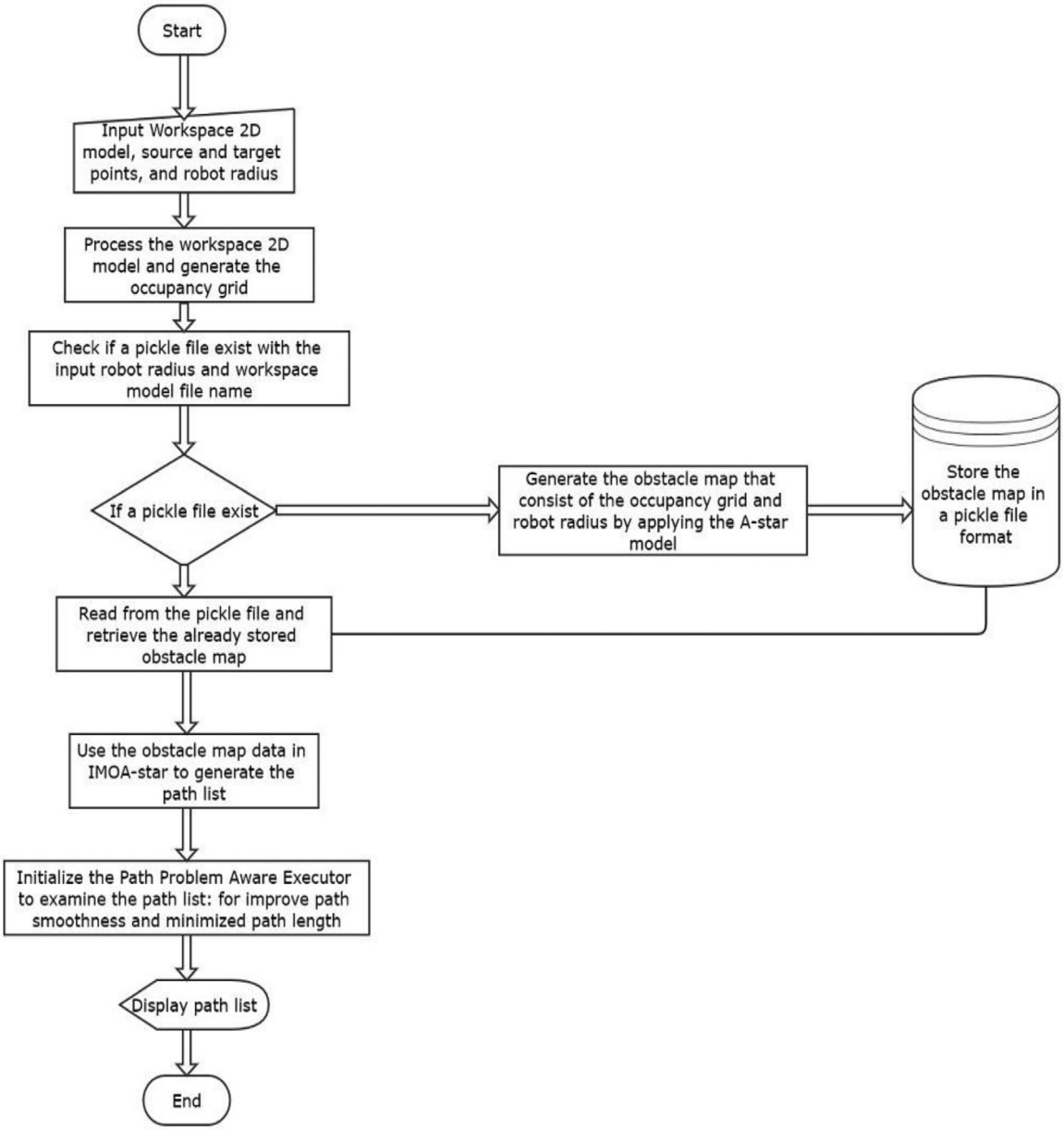

Fig. 1. Flowchart for the proposed IMOA-star.

$P_1 = (r_1, c_1)$and $P_2 = (r_2, c_2)$ is specified by [21–22]:

$$d(P_1, P_2) = \sqrt{(r_1, c_1)^2 + (r_2, c_2)^2}. \tag{2}$$

Since the entire path is made up of a list of points, the length of the entire path is specified by [21].

$$Path_{length} = \sum_{i=1}^{n-1} d(P_i, P_{i+1}) \tag{3}$$

Where$n$is the size of the list that represents the path:$P_1$and$P_{i+1}$are the$ith$and the $(i+1)th$items of the list of points.

Path smoothness objective

The goal is to keep the path as straight as feasible by keeping the angle between succeeding segments of the path as near 180° as possible. As a result, the robot follows a path with fewer turns and consumes less energy. The path smoothness

model as presented in [21] is:

$$Path_{smoothness} = \sum_{i=1}^{n_\alpha} (180^o - \alpha_i) \tag{4}$$

Where$n_\alpha$is the number of path angles,$\alpha_i$is the value of theith angle between each path segement of the path measured in degree from$0^o - 180^o$. The objective of the IMOA-star algorithm is to minimize$(180^o - \alpha_i)$.

The following is the operation process for the IMOA-star:

Workspace model

It is necessary to create a 2D model of the workspace to solve the path planning problem within it. This representation provides the path planning algorithm with the information it needs to calculate the traversal path. Inkscape-0.92.1, open-source software for creating picture files on the Linux operating system, was used to build the 2D floor plan model for the workspace. The workspace, walls, and static obstacles are all depicted in pixel units. The workspace is 7120 cm $\times$ 9490 cm in size, and the model is 7120 px $\times$ 9490 px in resolution. The black color represents walls and obstacles in the workspace, whereas the white hue represents open space. The extension ".png" is used to save this image file. The model resolution is scaled down by ten before exporting the 2D file. This is useful for a large workspace and speeds up the calculation.

Input phase

The 2D model is imported into the IMOA-star with the file extension ".png" in Python 3.8.3 64-bit. The radius of the robot, as well as the start and target points, are then entered.

Run IMOA-star

The IMOA-star generates a path list from a source location to a destination point. The IMOA-star determines whether the free space between the start and target points is greater than the radius of the robot. If this condition is met, IMOA-star generates the obstacle map; otherwise, a void list is returned. The obstacle map created at the IMOA-star first run is a Python object structure. This Python object structure is serialized (converted to bytes) and saved as a pickle file. The map picture filename pickle_rr_robot radius in cm is the storage format. The path list from the start point to a goal point is then generated. However, before the IMOA-star prints the path list, the included path problem-aware executor analyzes the path to shorten it and improve its smoothness. The pseudo-code for the IMOA-star. *Agorithm:* IMOA-star

Inputs: The source location $(\boldsymbol{x}, \boldsymbol{y})$, target position $(\boldsymbol{x}, \boldsymbol{y})$ points, static obstacle points $(\boldsymbol{x}, \boldsymbol{y})$, workspace 2D model, and robot radius (cm)

Output: Path list

Process: Process the workspace 2D model and generate the occupancy grid

1. If a pickle file exists with the input robot radius and workspace model file name:
   Read from the pickle file and retrieve the already stored obstacle map
   Else Generate the obstacle map that consist of the occupancy grid and robot radius by applying the A-star mathematical model and Store the obstacle map in a pickle file format
   End
2. Use the obstacle map data in IMOA-star to generate the path list
   Do
   Initialize the Path Problem Aware Executor to examine the path list:
   Generate a new empty path list = []
   Append source location as first element in the new path list [$\boldsymbol{P_{i-1}}$]
   For $\boldsymbol{i}$ in range $\boldsymbol{P_{i-1}}$ to $\boldsymbol{P_{ith}}$: draw a straight line from $\boldsymbol{P_{i-1}}$ and $\boldsymbol{P_i}$
   If straight line from $\boldsymbol{P_{i-1}}$ to $\boldsymbol{P_i}$ does not cross the obstacle map
   draw a straight line from $\boldsymbol{P_{i-1}}$ and $\boldsymbol{P_{i+1}}$
   else append the last known point from $\boldsymbol{P_{i-1}}$ that returns true to the new path list
   End
   End
   While target position is attainable
   return optimal path list

Path segment length and orientation measurement

To measure the path segment length and orientation between two coordinate points $A(x_o, y_o)$ and$B(x_1, y_1)$Eqs. (5) and 6 is applied.

Table 2
Cases and Parameters used to compare the IMOA-star and the traditional A-star.

Case	Start Point (cm)(x, y)	New Obstacle Vertex Point	Goal point (cm)(x, y)
1	(70, 8200)	Nil	(4900, 370)
2	(70, 8200)	px1 = [348, 354, 359, 355]py1 = [850, 861, 866, 877]	(4900, 370)
3	(3800, 8800)	Nil	(6800, 3400)
4	(4900, 460)	Nil	(3700, 8700)

The distance between$A(x_o, y_o)$and $B(x_1, y_1)$is given by [22]:

$$d = \sqrt{(x_1 - x_o)^2 + (y_1 - y_o)^2} \tag{5}$$

$$\alpha = \tan^{-1}\left(\frac{y_1 - y_0}{x_1 - x_o}\right), \ \left(0^o \leq \alpha \leq 360^o\right) \tag{6}$$

where $d$is the distance between the two points, $\alpha$is the orientation bearing of the final point$B$from an initial point$A$, x_0, y_0 and x_1, y_1 are the 2D coordinate location for points A and B respectively.

Result and discussion

In the case scenario, this section displays the simulations and results of a comparison of the IMOA-star and the traditional A-star. The IMOA-star and traditional A-star were implemented using Python 3.8.3 64-bit ('anaconda3′: Virtualenv). Math, matplotlib. pyplot, NumPy, Pandas, and time libraries are among the imported libraries. Modules. ogmap was built to produce the pickle file, modules. astar is the A-star function and modules. func is the path-problem-aware executor. The Python program returns the processing time, path segment angles, the number of points to be traversed, and path length. The entire code was tested on a computer with an Intel (R) Core (TM) i7–8850H CPU running at 2.60 GHz, 2592 MHz, 6 cores, and 12 logical processors (s). The simulation instances used are shown in Table 2, and the results of the comparison between the IMOA-star and the traditional A-star are shown in Table 3.

From Table 3, Both the traditional A-star and the IMOA-star processes take the same amount of time in Case 1. This is because in Case 1, the pickle file for the workspace map and robot radius was only recently created in IMOA-star. Even with the addition of the new obstacle in Case 2, the IMOA-star shows a significant reduction in process time and path length, as well as improved path smoothness. When the start and target points are changed in Cases 3 and 4, the IMOA-star still shows a significant reduction in process time and path length, as well as improved path smoothness, because the pickle file is saved and the obstacle map is not recreated, as in traditional A-star. The IMOA-star reduced path length by an average of 1.58% and processing time by an average of 99.98% in all four cases. The IMOA-star reduces the traditional A-star's random point to an average of approximately 83.45% and minimizes $(180^o - \alpha_i)$ to an average of approximately 24.75° for the four cases. This is advantageous for non-holonomic mobile robots since every point has the potential to be a turn coordinate, and the minimum $(180^o - \alpha_i)$ ensures that the turn direction is closer to 180°.

According to the processing time output graph in Fig. 2, for Case 1, the IMOA-star and traditional A-star both achieved a processing time of approximately 22,717.3 s (6 hrs.: 32 min). Because, in Case 1, the IMOA-star is just creating the pickle file for the workspace map and robot radius. However, with a change in start and goal positions, as well as the addition of a new static obstacle in Case 2, the IMOA-star process time for Cases 2–4 is approximately 5 s on average. The traditional A-star, on the other hand, spends an average of 6 h and 7 min. The IMOA-star reduced processing time in cases 2–4 by 99.98%. This demonstrates the importance of the pickle file created in Case 1.

The average angle between path segments (α) for IMOA-star for Cases 1–4 is about 152°. This is closer to 180° than the traditional A-star's 135° offering. As a result, the IMOA-star's path is smoother, and the model in Eq. (4) is minimized to an average of roughly 24.75%, compared to 45% for the A-star, as shown in Fig. 3 path segment angle output.

Fig. 4 shows that for Cases 1–4, IMOA-star presented an average of 19 points, while A-star presented 148 points. The number of random points has been reduced by 83.45% as a result of the IMOA-star. Because these points represent the location of orientation changes, the robot will use an IMOA-star to perform fewer rotations, reducing the torque required by the wheel DC motor and extending battery life.

According to the path length output in Fig. 5, the IMOA-star reduced the path length for Cases 1–4 by 1.74%, 2.01%, 1.29%, and 1.28%, respectively, for a total reduction of roughly 1.58% for these test cases. This demonstrates how the IMOA-star minimizes Eq. (2) and Eq. (3), causing the robot to travel less distance and take less time to complete its circle.

Conclusion

The design of IMOA-star for mobile robot path planning in a large workspace is discussed in this paper, as well as its implementation in Python 3.8.3 64-bit ('anaconda3′: Virtualenv). The IMOA-star method is a viable alternative to the traditional A-star method for path planning in a large workspace. Four test cases were used to assess the performance of

Table 3
Simulation results for the IMOA-star and traditional A-star in the comparison study.

Case	Traditional A-star Algorithm				IMOA-star Algorithm				Comparison			
	Process Time (*sec*)	Number of Points	Path Length (cm)	Path Segment angle ($180^o - \alpha_i$)minimized	Process Time (*sec*)	Number of Points	Path Length (cm)	Path Segment angle ($180^o - \alpha_i$)minimized	% decrease in processing time	% decrease in path length	% increase in Path Smoothness	% decrease in the random point
1	22,717.3	209	13,968.63	45	22,717.3	21	13,726.02	27.63	0	1.74	38.6	89.95
2	23,847.1	221	13,968.62	45	7.45	22	13,687.24	26.25	99.97	2.01	41.67	90.04
3	23,883.8	60	8288.28	45	2.83	20	8180.6	23.11	99.99	1.29	48.64	66.67
4	24,722.2	101	9268.84	45	4.51	13	9150.48	22	99.98	1.28	51.11	87.13

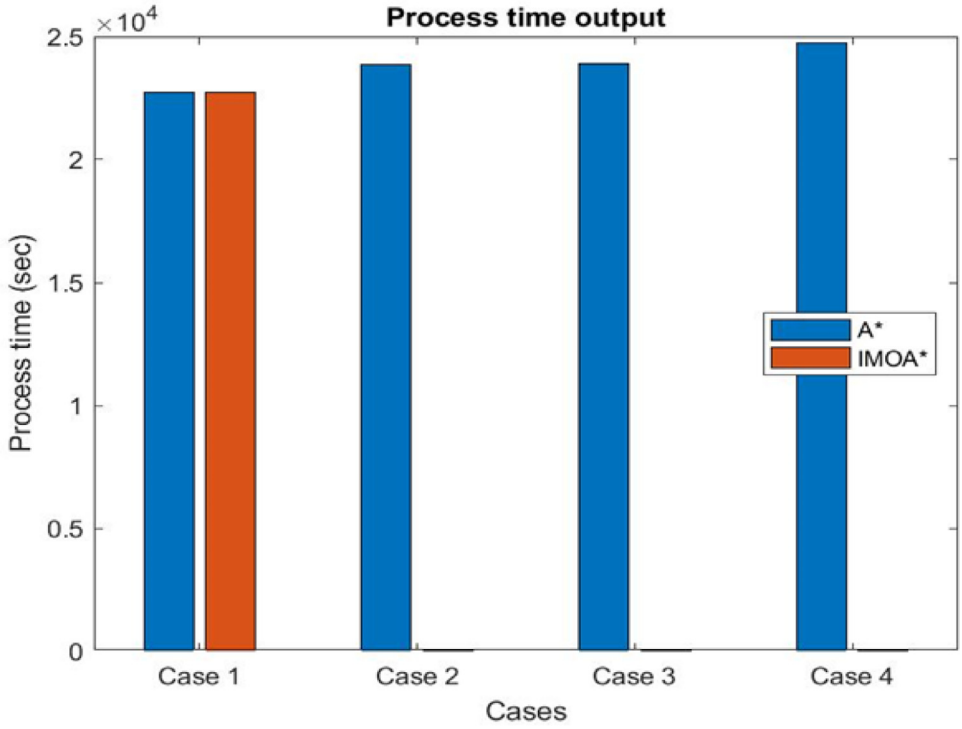

Fig. 2. IMOA_star and traditional A_star graphical images for process time output results and comparison.

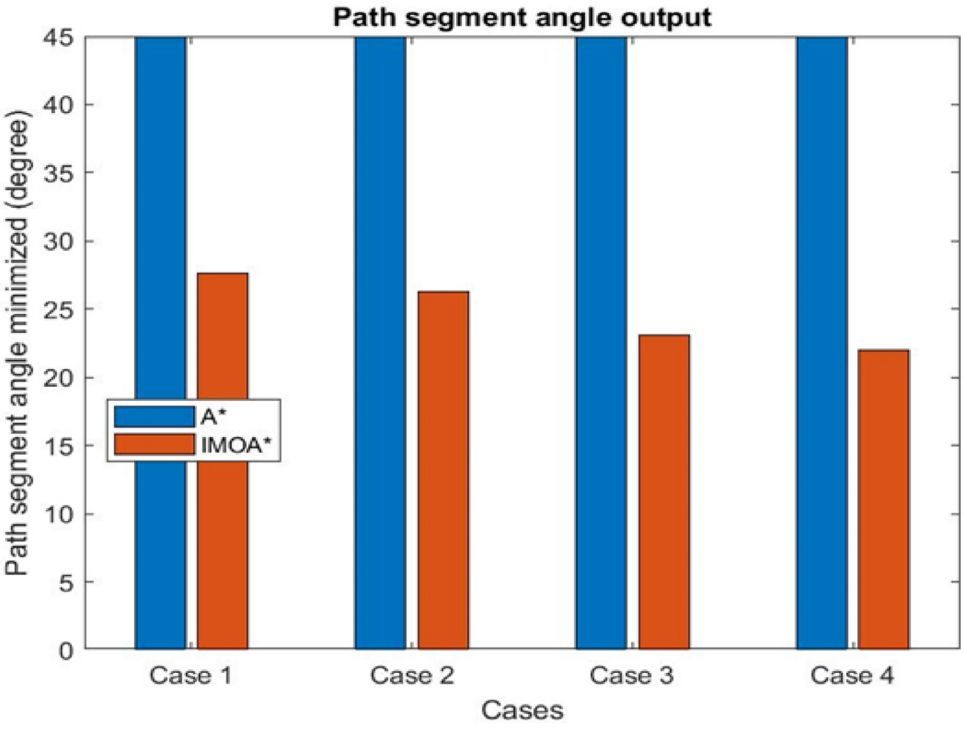

Fig. 3. IMOA_star and traditional A_star graphical images for path segment angle output results and comparison.

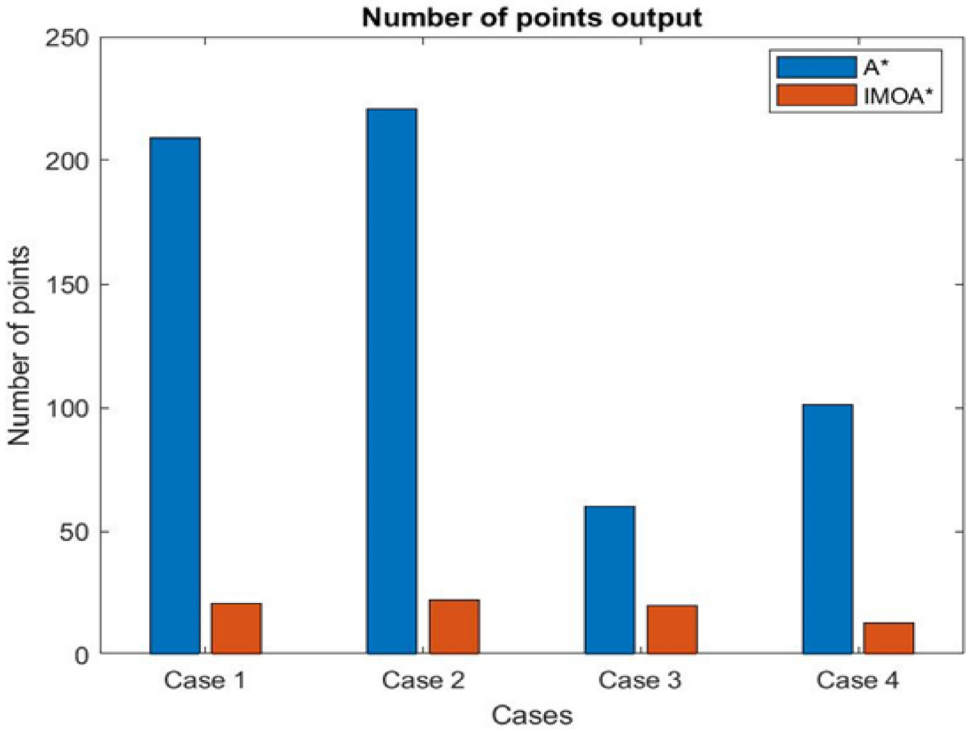

Fig. 4. IMOA_star and traditional A_star graphical images for the number of points output results and comparison.

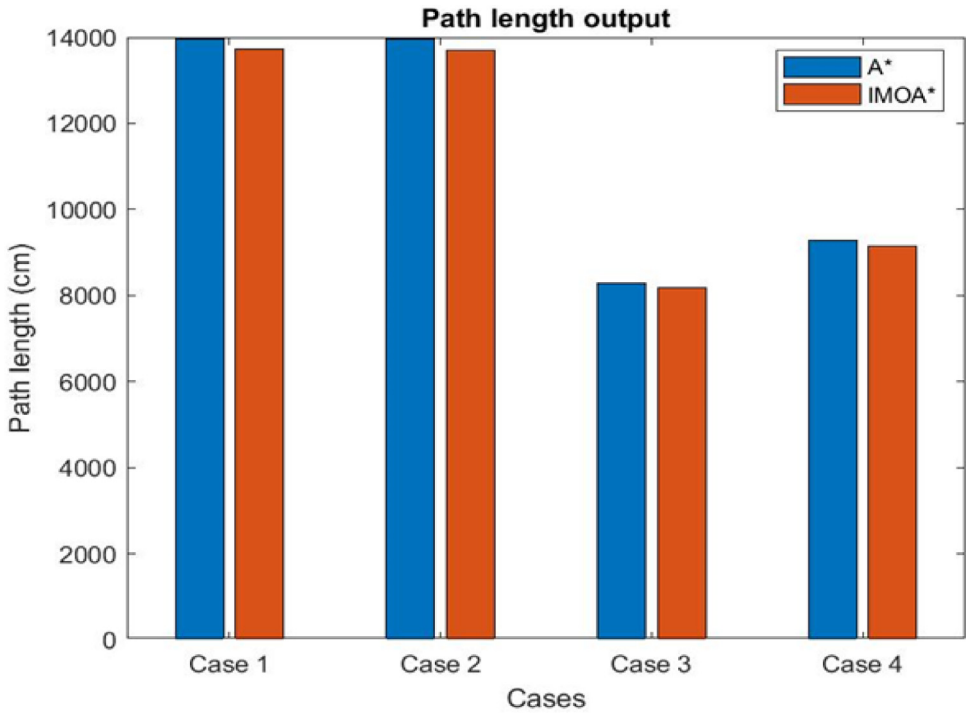

Fig. 5. IMOA_star and traditional A_star graphical images for path length output results and comparison.

the IMOA-star algorithm in terms of algorithm process time, path smoothness, number of points, and path length, and the results were compared to the traditional A-star algorithm. The following are the conclusions drawn from the findings:

1 Despite the change in start and target points, as well as the addition of a new obstacle in the workspace, the average process time for IMOA-star after the pickle is generated in Case 1 is approximately 99.98% less than the traditional A-star.
2 In comparison to the traditional A-star, the IMOA-star improved path smoothness by presenting path segment angles closer to 180o, and reduced Eq. (4) by an average of 24.75%, compared to 45% for the A-star.
3 The IMOA-star reduced the path length by an average of 1.58% for each of the four test cases when compared to the traditional A-star for the identical test cases. As a result, the IMOA-star reduced Eq. (2) and Eq. (3) to their simplest form.
4 The IMOA-star reduces the number of points for the four test cases by 83.45% when compared to the traditional A-star. When compared to the traditional A-star, the reduction in random points causes the robot to perform significantly fewer turns, reducing torque on the wheel DC motors and improving battery life.

The study discovered that the IMOA-star outperforms the traditional A-star in general, and it may thus be considered a viable alternative for mobile robot path planning in a large workspace.

Funding

No funding from an external source was received by the authors.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Oluwaseun Opeyemi Martins: Conceptualization, Methodology, Software, Writing – original draft. **Adefemi Adeyemi Adekunle:** Data curation. **Olatayo Moses Olaniyan:** Visualization, Investigation. **Bukola Olalekan Bolaji:** Supervision, Writing – review & editing.

References

[1] A. Hossain, I. Ferdous, Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, Rob. Auton. Syst. (2014) 1–5.
[2] H.T. Nguyen, H.L. Xuan, Path planning and obstacle avoidance approaches for mobile robot, Int. J. Comput. Sci. Issues (2016) 3–4.
[3] S.A. Gunawan, N.P.P. Gilang, A.I.W.B. Cahyadi, Y.C.H. Yuwono and O. Wahyunggoro, "Smoothed A-star algorithm for nonholonomic mobile robot path planning," in International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2019.
[4] A. Hidalgo-Paniagua, M.A. Vega-Rodríguez, J. Ferruz, N. Pavón, MOSFLA-MRPP: multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning, Eng. Appl. Artif. Intell. 44 (2015) 123–136.
[5] S. Benders and S. Schopferer, "A line-graph path planner for performance constrained fixed-wing uavs in wind fields," in International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 2017.
[6] F. DuchoĚ, A. Babineca, M. Kajana, P. BeĚoa, M. Floreka, T. Ficoa, L. Jurišicaa, Path planning with modified A star algorithm for a mobile robot, in: Procedia Engineering Modelling of Mechanical and Mechatronic Systems MMaMS 2014, 96, 2014, pp. 59–69.
[7] M.S. Ganeshmurthy, G.R. Suresh, Path planning algorithm for autonomous mobile robot in dynamic environment, 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, India, 2015.
[8] D. Ferguson, M. Likhachev, A. Stentz, A Guide to Heuristic-based Path Planning, American Association for Artificial Intelligence, 2005.
[9] J.M. Yang, C.M. Tseng, P.S. Tseng, Path planning on satellite images for unmanned surface vehicles, Int. J. Naval Arch. Ocean Eng. 7 (1) (2015) 87–99.
[10] K.T.H.T.T. Izumi, Nonholonomic control considering with input saturation for a mobile robot, 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Tsukuba, Japan, 2016.
[11] T.R. Schafle, A. Tokui, N. Uchiyama, A hybrid systems approach with input-output linearization for automotive parking control of a nonholonomic mobile robot, 3rd International Conference on Advanced Robotics and Mechatronics (ICARM), Singapore, Singapore, 2018.
[12] M. Lin, K. Yuan, C. Shi, Y. Wang, Path Planning of mobile robot based on improved A* Algorithm, 29th Chinese Control and Decision Conference (CCDC), 2017.
[13] J. Yu, J. Hou, G. Chen, Improved safety-first a-star algorithm for autonomous vehicles, 5th International Conference on Advanced Robotics and Mechatronics (ICARM), 2020.
[14] H-m. Zhang, M-l. Li, Rapid path planning algorithm for mobile robot in dynamic environment, Adv. Mech. Eng. 9 (12) (2017) 1–12.
[15] E. Shang, B. Dai, Y. Nie, Q. Zhu, L. Xiao, D. Zhao, An improved A-star based path planning algorithm for autonomous land vehicles, Int. J. Adv. Rob. Syst. (2020) 1–13.
[16] C. Ju, Q. Luo and X. Yan, "Path Planning using artificial potential field method And A-star fusion algorithm," Global Reliability and Prognostics and Health Management (PHM-Shanghai), Shanghai, China, pp. 1–7, 2020.
[17] C. Liang, X. Zhang, Y. Watanabe, Y. Deng, Autonomous collision avoidance of unmanned surface vehicles based on improved A star and minimum course alteration algorithms, Appl. Ocean Res. 113 (2021) 1–11.
[18] K. Jeddisaravi, R.J. Alitappeh, F.G. Guimari'ies, Multi-objective mobile robot path planning based on A* search, 6th International Conference on Computer and Knowledge Engineering (ICCKE 2016), Ferdowsi University of Mashhad, 2016.
[19] R. Yang, L. Cheng, Path planning of restaurant service robot based on A-star algorithms with updated weights, 12th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2019.
[20] G. Qing, Z. Zhang, X. Yue, Path-planning of automated guided vehicle based on improved Dijkstra algorithm, 29th Chinese Control and Decision Conference (CCDC), Chongqing, China, 2017.

[21] A. Hidalgo-Paniagua, M.A. Vega-Rodríguez, J. Ferruz, N. • Pavón, Solving the multi-objective path planning problem in mobile robotics with a firefly--based approach, Soft Comput. 21 (4) (2017) 949–964.
[22] X.Y. Wang, G.X. Zhang, J.B. Zhao, H.N. Rong, F. Ipate, R. Lefticaru, A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning, Int. J. Comput. Commun. Control 10 (5) (2015) 732–745.