EVOLUTION IN ELECTRICAL AND ELECTRONIC ENGINEERING
e-ISSN: 2756-8458

EEEE

Vol. 5 No. 1 (2024) 332-340
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Improved Particle Swarm Optimization (IPSO) Based Mobile Robot Navigation for Path Planning

Kogileswaran Naidu[1], Rohaida Mat Akir[1]*

[1] *Department of Electrical Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, Johor, MALAYSIA.*

*Corresponding Author: rohaida@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2024.05.01.042

Article Info

Received: 11 January 2024
Accepted: 15 March 2024
Available online: 30 April 2024

Keywords

Particle Swarm Optimization (PSO), A* algorithm, Dijkstra algorithm

Abstract

This research delves into the advancement of mobile robot path planning within indoor environments, leveraging the Particle Swarm Optimization (PSO) algorithm. The significance of mobile robots spans diverse applications, ranging from surveillance and exploration to rescue operations and industrial automation. The objective of this study is to formulate a resilient and efficient path planning methodology tailored for mobile robots navigating through unknown and dynamic environments. Integration of the PSO algorithm with renowned path planning algorithms like A* and Dijkstra is undertaken to optimize navigation by refining obstacle avoidance, minimizing path length, and enhancing overall efficiency within real-time constraints. The investigation encompasses a comprehensive exploration of mobile robot navigation principles, an in-depth analysis of obstacle avoidance efficacy, and the development of an algorithm adept at navigating with reduced turns. The implementation utilizes the Python programming language to craft path planning and navigation algorithms. Ultimately, this research aims to contribute a dependable and optimized solution for path planning, fostering autonomous navigation proficiency in intricate indoor settings.

1. Introduction

The introduction provides a comprehensive overview of the burgeoning field of mobile robotics, underscoring its remarkable growth and widespread applications across diverse sectors. Mobile robots, endowed with versatile capabilities, are increasingly substituting humans in tasks such as surveillance, planetary exploration, patrolling, emergency rescue operations, industrial automation, and more. The intricate process of mobile robot navigation involves four stages: perception, localization, cognition and path planning, and motion control. The significance of safe path planning is emphasized, necessitating obstacle identification and avoidance as robots traverse from the starting point to their destination, meeting specific criteria like distance and smoothness of the path. Two distinct types of navigation, global and local, cater to varying environmental familiarity levels. Path planning, a dynamic process, encompasses classical algorithmic methods (A*, Dijkstra, RTT, artificial potential field) and heuristic approaches (differential evolution, genetic algorithm, ant colony algorithm, artificial fish swarm algorithm, and PSO). The Particle Swarm Optimization (PSO) algorithm stands out for its exceptional search capability, rapid convergence speed, and efficiency. Its reliance on minimal hyper-parameters and simplicity without the need for advanced knowledge contribute to its widespread adoption in both practical applications and theoretical studies of mobile agent path planning. The project's central focus is on developing a robust path planning method for mobile robots operating in unknown and dynamic environments. This involves integrating the PSO algorithm with traditional path planning algorithms such as A* and Dijkstra. The objectives include enhancing obstacle avoidance, reducing path length, and optimizing navigation efficiency, all while considering real-time constraints. The overarching goal is to contribute significantly to the field of mobile robotics, providing a reliable and optimized solution for autonomous navigation in complex environments. Further, the research aims to delve into the realm of mobile robot path planning using the Python programming language. The proposed approach involves a meticulous exploration of algorithms dedicated to charting the shortest paths within unknown indoor environments. An in-depth analysis of obstacle avoidance performance and the optimization of smoothness, specifically reducing the number of turns required to reach the destination, are key components of the research objectives. The scope extends to addressing limitations inherent in path planning for mobile robot navigation within urbanized areas. The project's multi-faceted approach encompasses fundamental concepts of navigation planning, the implementation of path planning using the PSO technique, and the development of an Improved PSO method incorporating A* and Dijkstra algorithms. Throughout this process, the Python programming language serves as the primary tool, facilitating the implementation of path planning algorithms and contributing advancements to the field of mobile robot navigation.

2. Classical Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based stochastic optimization algorithm that was proposed by Kennedy and Eberhart in 1995 [1]. It is inspired by the collective behavior of bird flocking or fish schooling, where individuals coordinate their movements to achieve a common goal. PSO utilizes a swarm of particles, each representing a potential solution, to explore the search space and find the optimal solution. Each particle adjusts its position based on its own experience and the collective experience of the swarm. The algorithm iteratively updates the velocity and position of each particle until a termination condition is met. The classical PSO is simple to implement and has few adjustment parameters when it is used in the path planning method [2]. However, this classic PSO is prone to poor searchability, falls into the optimal local solution, reduced particle diversity, low convergence precision, and low accuracy of path planning. Equation (1) shows the PSO algorithm. The parameter W is the inertia weight, and it is a positive constant. It is for balancing global search known as exploration (when higher values are set) and local search as exploitation (when lower values are set).

$$V_i^{t+1} = W.V_i^t + c_1 U_1^t\left(P_{b_1}^t - P_i^t\right) + c_2 U_2^t\left(g_{b_1}^t - P_i^t\right) \quad \ldots\ldots\ldots\ldots (1)$$

2.1 Comparison between existing Path Planning Algorithms

There are various types of algorithms that exist for path planning. Among those algorithms PSO, A* and Dijkstra are the three most effective algorithms for path planning. So, by combining the PSO algorithm with A* and Dijkstra algorithms, more effective path planning algorithms can be created. Table 1 shows types of algorithms that are commonly used for path planning and table 2 shows types of algorithms that are exist in PSO.

Table 1 *Literature Review for Path planning*

Author	Types of Algorithms	Explanations
O.A Gbadamosi and D. R. Aremu, 2020 [3]	Dijkstra	This algorithm alone can't use within dynamic environments and need to handle negative edges.
S. A. Gunawan, G. , A. Cahyadi2019 [4]	A*	With the development of Artificial Intelligence, the A* algorithm has been improved robot path planning.
Song Yong, Gao Tengteng, L. G. (2020). [5]	Artificial Potential Field (APF)	Provide effective motion planning method but the robot is easily caught at local minimum before achieving its goal.
F.Qu, W. Yu, K. Xiao and C. Liu,2022 [6]	Ant Colony Optimization	Has defects of low search efficiency and slow convergence speed.
S. Gao and Y. Wen, 2018 [7]	Artificial Fish algorithm (AFO)	Slow convergence in the late time and limited precisions.
L. Zhang, Y. Zhang and Y. Liu, 2021 [8]	Particle Swarm Optimization (PSO)	Less parameters adjustments and fast search speed.

Table 2 *Literature Review for PSO Algorithm*

Author	Types of Algorithms	Explanations
X. Li, D. Wu, J. He, M. Bashir and M. Liping, 2020 [9]	Classical PSO	Simple to implement but prone to poor searchability, low accuracy of path planning
B. Song, Z. Wang, L. Xu, L. Zou and F.E Alsaadi, 2019 [10]	Modified Particle Swarm Optimization (MPSO)	Can obtain a smooth path but does not improve the efficiency of the particle diversity.
P. Chen, Q. Li, C. Zhang, J. Cui and H. Zhou, 2019 [11]	Ant Colony Particle Swarm Optimization (AC-PSO)	Particle global search efficiency is restricted.
L. Zhen Du, S. Ke , Z. Wang, J. Tao, L. Yu and H. Li, 2019 [12]	Genetic Particle Swarm Optimization (GA-PSO)	Number of parameters are lot
M. N. A. Wahab, C.M. Lee, M.F Akbar and F.H Hassan, 2020 [13]	Fringe Search Particle Swarm Optimization (PSOFS)	Shorter, smoother, and safer pathways is created in unfamiliar interior situations.
X. Li, D. Wu, J. He, M. Bashir and M. Liping, 2020 [14]	Improved Particle Swarm Optimization (IPSO)	IPSO can yield better optimum outcome with fewer iteration steps

Based on Table 2 among the path planning algorithms A* and Dijkstra shows better performance in path planning based on the authors. While based on Table 3, among the types of PSO algorithms, IPSO yields better performance in path planning. So, combining the A* or Dijkstra with PSO algorithms can produce better IPSO algorithm for robot path planning.

3. Methodology

The Improved Particle Swarm Optimization (IPSO) based mobile robot navigation for path planning, comparing the A* algorithm and Dijkstra algorithm using MATLAB, involves several steps. First, the environment is modelled using a grid-based representation, where obstacles and target positions are defined. Then, the A* algorithm and Dijkstra algorithm will be implemented to find the optimal paths from the robot's initial position to the target position. The algorithms use heuristic functions and weight factors to determine the most efficient paths. Next, the IPSO algorithm is applied to optimize the paths obtained from both algorithms, by adjusting the particle swarm optimization parameters. The IPSO algorithm enhances the exploration and exploitation capabilities of the swarm, improving the path planning performance. Finally, the paths generated by both algorithms, before and after the IPSO optimization, are compared based on criteria such as path length, execution time, and smoothness. This methodology allows for a comprehensive evaluation of the A* and Dijkstra algorithms in terms of their performance and efficiency for mobile robot navigation and path planning when combining with PSO.

3.1 Flowchart

As shown in Fig.1 firstly it defines a Particle class representing particles in the PSO algorithm, a cost function to calculate the distance between a position and a goal, and a function to check if a point is within obstacles. The PSO function initializes particles, updates their positions and velocities based on a PSO update rule, and handles obstacle avoidance. The code also includes functions to generate a continuous Bezier curve passing through path waypoints and to visualize the path with Matplotlib. An example scenario is provided, where a PSO algorithm is used to find the optimal path from a start position to a goal while avoiding specified obstacles. The script prints the final position and visualizes the path with obstacles using Matplotlib. Adjustments to grid size, obstacles, and parameters can be made for different scenarios.

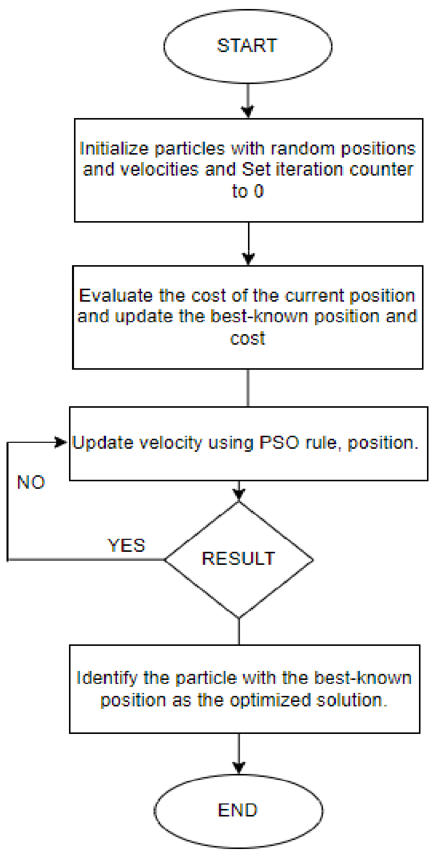

START
Initialize particles with random positions and velocities and Set iteration counter to 0
Evaluate the cost of the current position and update the best-known position and cost
Update velocity using PSO rule, position.
NO
YES
RESULT
Identify the particle with the best-known position as the optimized solution.
END

Fig. 1 *Flowchart of Implementation of Algorithms combinations*

3.1 Implementation of PSO

The Particle Swarm Optimization (PSO) algorithm is a population-based optimization technique where a group of particles explores a multidimensional solution space to find the optimal solution. Each particle's position in space is continuously adjusted based on its own historical best-known position) (particle.best_position) and the global best-known (best_particle.best_positon). The velocity of each particle is updated using the inertia weight (w), acceleration constants (c1&c2) and random values (r1&r2). The Particle Swarm Optimization (PSO) algorithm is characterized by several key parameters. The swarm size (num_particles) denotes the number of particles in the population, influencing the thoroughness of exploration. The number of iterations (max_iterations) determines how many times particles update their positions, impacting the convergence behavior. The inertia weight (w) balances the impact of previous velocities on current velocities, crucial for exploration and exploitation trade-off. The cognitive coefficient (c1) emphasizes a particle's historical best-known position, fostering local exploitation, while the social coefficient (c2) guides movement based on the global best-known position, encouraging global exploration. These parameters, such as the size of the swarm and coefficients, play pivotal roles in tailoring the PSO algorithm's performance for specific optimization tasks by adjusting the trade-off between exploration and exploitation. Default values for these parameters are set. Table 3 shows the parameters that were set for the exploration and exploitation of the environment.

Table 3 *Parameters for Algorithms*

Parameters	Value
Number of particles	100
Number of Iteration	500
Inertia Weight	0.5
Cognitive Coefficient	1.5
Social Coefficient	1.5

3.2 Dijkstra Algorithm and Particle Swarm Optimization

In the Particle Swarm Optimization with Dijkstra (PSOD) algorithm, the integration of Dijkstra's algorithm and Particle Swarm Optimization (PSO) is evidenced through a combination of Dijkstra's path-finding approach and

PSO's dynamic optimization capabilities. The core of the integration lies in the update of particle positions. Dijkstra's algorithm, as implemented in the Dijkstra function, calculates the shortest path from the current particle position to the goal node. The equation tentative_cost = current.cost + 1 represents the tentative cost of reaching a neighbor node in Dijkstra's algorithm, and the comparison if tentative_cost < neighbor.cost ensures the selection of the most cost-efficient path. The PSO update rule involves equations such as particle.vx = w * particle.vx + c1 * r1 * (particle.best_position[0] - particle.x) + c2 * r2 * (goal.x - particle.x), where w, c1, and c2 are adjustable parameters, and r1 and r2 are random factors. This equation guides particles to update their velocities based on both their personal best-known position and the global best-known position (the goal node), balancing exploration and exploitation. The synergy of these equations harmonizes the global search capabilities of PSO with the local optimization strengths of Dijkstra, resulting in an effective and adaptable PSOD algorithm for path planning in complex environments.

3.3 A * Algorithm and Particle Swarm Optimization

The combination of A* and PSO involves applying A* to each particle's current position within the 2D grid, where A* computes a local optimal path considering obstacles. The PSO algorithm concurrently optimizes particle movements by updating their velocities and positions based on inertia weight (w), acceleration constants (c1 & c2) and random values (r1 & r2). The velocity update equations incorporate the particle's best-known position and the global best-known position, guiding particles towards promising regions. After updating velocities, A* is used to find local optimal paths for each particle, and if a valid path is obtained, the particle's position is updated accordingly. This iterative process continues for a predefined number of iterations, aiming to combine A*'s path-finding precision with PSO's exploration capabilities to discover an optimized path through a dynamic and obstacle-filled environment. The equations (2) govern the velocity and position updates are as follows.

$$particle.vx = w \times particle.vx + c1 \times r1 \times (particle.best_{position[0]} - particle.x) + c2 \times r2 \times (goal.x - partilce.x)$$
$$particle.vy = w \times particle.vy + c1 \times r1 \times (particle.best_{position[1]} - particle.y) + c2 \times r2 \times (goal.y - partilce.y)$$
$$particle.y = particle.y + particle.vy$$
$$particle.x = particle.x + particle.vx \quad \ldots\ldots\ldots\ldots (2)$$

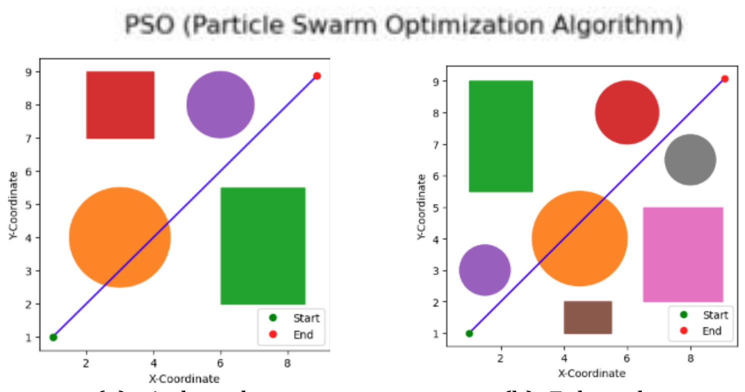
Python programming code Particle Swarm Optimization with A* Algorithm (PSOD) integrates the A* (A-star) pathfinding algorithm with the Particle Swarm Optimization (PSO) technique to navigate a 2D grid with obstacles from a designated start node to a goal node. The A* algorithm efficiently explores nodes while considering obstacles, and the PSO algorithm refines the solution by updating particle positions based on their best-known positions and velocities. The code employs modular structures, including a `Node` class for representing grid nodes, various helper functions for A* pathfinding, a `Particle` class for PSO, and visualization functions using Matplotlib. The example usage demonstrates the combined A* with PSO approach on a specified grid, showcasing the final position and visualizing the discovered path while considering obstacles. The flexibility and modularity of the implementation allow for easy experimentation and adaptation.

4. Results and Discussions

The output shows the Particle Swarm Optimization (PSO) algorithm. In the example scenarios, the Particle Swarm Optimization (PSO) algorithm navigates a 2D environment with obstacles represented as circles and rectangles. Based on the output obtained in Fig. 2, Fig. 3 and Fig. 4 with all the four and seven obstacle scenario included with a circular obstacle with a center at (3.5, 4) and a radius of 1.5, a rectangular obstacle from (6, 2) to (8.5, 5.5), a square from (2, 7) to (4, 9), and another circle with a center at (6, 8) and a radius of 1. The other 7 obstacles scenario introduces additional circular, rectangular, and square obstacles. The starting point is set at (1, 1), and the goal is at (8, 8). The PSO algorithm aims to find an optimized path from the starting point to the goal while avoiding collisions with the specified obstacles. The effectiveness of the algorithm in finding the shortest path is influenced by obstacle representations, algorithm parameters, and environmental complexity.
Authors declare that there is no conflict of interests regarding the publication of the paper.

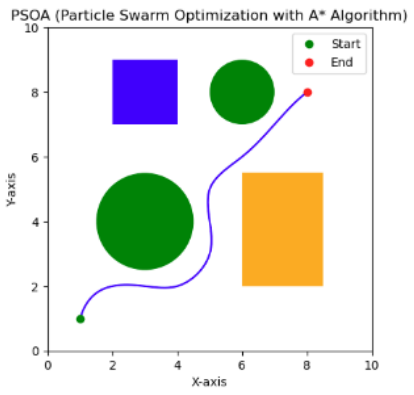
4.1 Particle Swarm Optimization (PSO)

Based on Fig.2 output displayed, the robot doesn't avoid the obstacle and hits it. The robot's failure to find a path and collision with obstacles in the provided PSO-based path planning algorithm can be attributed to several factors. The simplistic representation of obstacles as circles and rectangles may inadequately capture complex shapes, leading to collisions. The lidar sensor's range may be insufficient, causing the robot to detect obstacles too late. Poorly tuned algorithm parameters, such as the number of particles and cognitive/social coefficients in PSO, can hinder effective exploration of the solution space. Inertia weight and velocity update terms may be inadequately balanced, causing the robot to get stuck or move too rapidly.



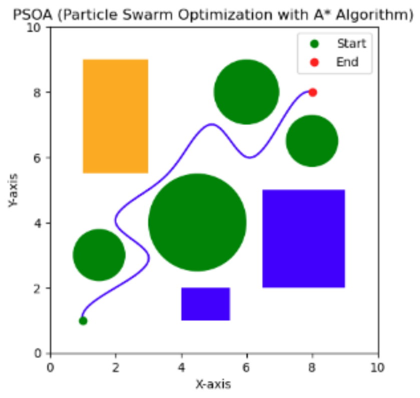
(a) 4 obstacles (b) 7obstacles

Fig. 2 *Particle Swarm Optimization (PSO) with (a) 4 obstacle and (b) 7 obstacles*



```
Path found: [(1, 1), (2, 2), (3, 2), (4, 2), (5, 3), (5, 4), (5, 5), (6, 6), (7, 7), (8, 8)]
Total distance traveled: 11.071067811865477
```

(a) 4 obstacles



```
Path found: [(1, 1), (2, 2), (3, 3), (2, 4), (3, 5), (4, 6), (5, 7), (6, 6), (7, 7), (8, 8)]
Total distance traveled: 12.727922061357859
```

(b) 7 obstacles

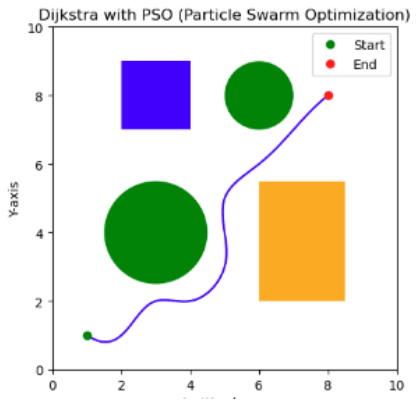
Fig. 3 *Particle Swarm Optimization with A* Algorithm with (a) 4 obstacles and (b) 7 obstacles*

4.2 Particle Swarm Optimization with A*

The output result of the PSOA (Particle Swarm Optimization with A*) provides a detailed insight into the path planning process. It starts by displaying the waypoints of the path discovered by the combined A* and PSO algorithms, outlining the coordinates through which the optimized trajectory navigates. The Total distance travelled was then determined, considering the Euclidean distance between consecutive waypoints, offering an alternative measure of the optimized path length. The graphical visualization enhances the understanding of the solution, presenting the grid layout with obstacles marked in distinctive colors such as green for circles, orange for rectangles, and blue for squares. The optimized path is illustrated as a smooth curve passing through waypoints, and the start and end points are highlighted by green and red dots, respectively. This visual representation aids in assessing the effectiveness of the combined algorithm in navigating around obstacles to reach the specified goal. If no viable path is found, the code appropriately communicates the absence of a solution. This combined algorithm was set to find the shortest path in two different environments which is one with 4 obstacles and another with 7 obstacles. This combined algorithm successfully found its endpoint without colliding the obstacles where total distance travelled in environment with 4 obstacles is 11.07 and for 7 obstacles environment is 12.73 as shown in Fig.3.

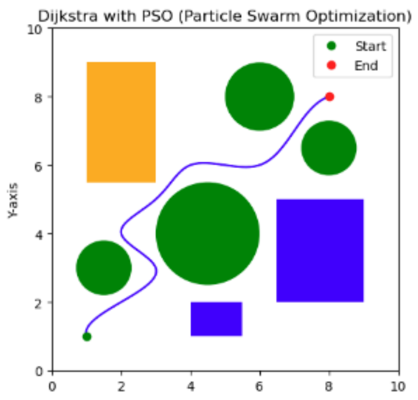
4.3 Particle Swarm Optimization with Dijkstra

```
Path found: [(1, 1), (2, 1), (3, 2), (4, 2), (5, 3), (5, 4), (5, 5), (6, 6), (7, 7), (8, 8)]
Total distance traveled: 11.071067811865477
```



(a) 4 obstacles

```
Path found: [(1, 1), (2, 2), (3, 3), (2, 4), (3, 5), (4, 6), (5, 6), (6, 6), (7, 7), (8, 8)]
Total distance traveled: 11.899494936611667
```



(c) 7 obstacles

Fig. 4 *Particle Swarm Optimization with Dijkstra Algorithm with (a) 4 obstacles and (b) 7 obstacles*

Based on Fig. 4 result of the PSOD (Particle Swarm Optimization with Dijkstra) algorithm represents the combination of Dijkstra's algorithm and Particle Swarm Optimization (PSO) to find a path from a defined start node to a goal node in a 2D grid with obstacles. The algorithm successfully discovers a path, which is visualized on a plot showing the grid, obstacles, and the trajectory represented by a continuous Bezier curve passing through the computed waypoints. The initial path, obtained through Dijkstra's algorithm, is refined using PSO to optimize the trajectory, considering the obstacles' influence. The transparency of obstacle shapes can be adjusted for better visualization. The program prints the found path and the total distance traveled, providing valuable insights into the effectiveness of the Dijkstra-PSO hybrid algorithm in navigating through obstacles to reach the goal. This combined algorithm was set to find the shortest path in two different environments which is one with 4 obstacles and another with 7 obstacles. This combined algorithm successfully found its endpoint without colliding the obstacles where total distance travelled in environment with 4 obstacles is 11.07 and for 7 obstacles environment is 11.90.

Based on Table 4, the PSO algorithm itself fails to create mapping in both environments while combination of PSOA and PSOD have value for shortest distance travelled for both type environments. For non-complex environment PSOA has smoother turns but same distance as PSOD. For complex environment PSOD has the shortest path travelled making it suitable combination with PSO for path planning navigation.

Table *4 Algorithm Performance Comparison in Path Planning Scenarios*

Algorithm	Distance Travelled Based on Number of Obstacles	
	4	7
PSO (Particle Swarm Optimization)	No Path Found	No Path Found
PSOA (PSO Combined with A*)	11.07	12.73
PSOD (PSO Combined with Dijkstra)	11.07	11.90

The findings of this research hold significant implications for the field of mobile robotics, particularly in the domain of path planning. The successful integration of the Particle Swarm Optimization (PSO) algorithm with established path planning algorithms like A* and Dijkstra demonstrates a promising approach to enhance the efficiency and robustness of mobile robot navigation in complex indoor environments. The optimized paths generated by the PSO algorithm contribute to improved obstacle avoidance, reduced path lengths, and overall enhanced navigation performance. This research not only expands our understanding of mobile robot navigation strategies but also provides a practical and applicable solution for real-world scenarios. The implications of these findings extend to various industries, including surveillance, exploration, and industrial automation, where autonomous mobile robots must navigate dynamically changing environments with precision and efficiency. The outcomes of this study paved the way for the development and deployment of more sophisticated and reliable autonomous mobile robots capable of seamlessly navigating through challenging indoor spaces.

Acknowledgement

This research is supported Faculty of Electrical and Electronic Engineering, University Tun Hussein Onn Malaysia (UTHM).

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] Slowik, A. (2011). Particle Swarm Optimization. The Industrial Electronics Handbook - Five Volume Set, May 2011. https://doi.org/10.1007/978-3-319-46173-1_2.

[2] Li, X., Wu, D., He, J., Bashir, M., & Liping, M. (2020). An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot. Journal of Control Science and Engineering, 2020. https://doi.org/10.1155/2020/3857894

[3] Gbadamosi, O. A., & Aremu, D. R. (2020). Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs. 2020 International Conference in Mathematics, Computer Engineering and Computer Science, ICMCECS 2020. https://doi.org/10.1109/ICMCECS47690.2020.240873

[4] Gunawan, S. A., Pratama, G. N. P., Cahyadi, A. I., Winduratna, B., Yuwono, Y. C. H., & Wahyunggoro, O. (2019). Smoothed a-star algorithm for nonholonomic mobile robot path planning. 2019 International Conference on

Information and Communications Technology, ICOIACT 2019, 654–658. https://doi.org/10.1109/ICOIACT46704.2019.8938467

[5] Wang Di, Li Caihong, Guo Na, Song Yong, Gao Tengteng, L. G. (2020). Local Path Planning of Mobile Robot Based on Artificial Potential Field. 3677–3682. https://doi.org/10.23919/CCC50068.2020.9189250

[6] Qu, F., Yu, W., Xiao, K., Liu, C., & Liu, W. (2022). Trajectory Generation and Optimization Using the Mutual Learning and Adaptive Ant Colony Algorithm in Uneven Environments. Applied Sciences (Switzerland), 12(9). https://doi.org/10.3390/app12094629

[7] Gao, S., & Wen, Y. (2018). An Improved Artificial Fish Swarm Algorithm and its Application. Proceedings - 17th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2018, Meici, 649–652. https://doi.org/10.1109/ICIS.2018.8466458

[8] Zhang, L., Zhang, Y., & Li, Y. (2021). Mobile Robot Path Planning Based on Improved Localized Particle Swarm Optimization. IEEE Sensors Journal, 21(5), 6962–6972. https://doi.org/10.1109/JSEN.2020.3039275

[9] Li, X., Wu, D., He, J., Bashir, M., & Liping, M. (2020). An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot. Journal of Control Science and Engineering, 2020. https://doi.org/10.1155/2020/3857894

[10] Song, B., Wang, Z., Zou, L., Xu, L., & Alsaadi, F. E. (2019). A new approach to smooth global path planning of mobile robots with kinematic constraints. International Journal of Machine Learning and Cybernetics, 10(1), 107–119. https://doi.org/10.1007/s13042-017-0703-7

[11] Chen, P., Li, Q., Zhang, C., Cui, J., & Zhou, H. (2019). Hybrid chaos-based particle swarm optimization-ant colony optimization algorithm with asynchronous pheromone updating strategy for path planning of landfill inspection robots. International Journal of Advanced Robotic Systems, 16(4), 1–11. https://doi.org/10.1177/1729881419859083

[12] Du, L. zhen, Ke, S., Wang, Z., Tao, J., Yu, L., & Li, H. (2019). Research on multiload AGV path planning of weaving workshop based on time priority. Mathematical Biosciences and Engineering, 16(4), 2277–2292. https://doi.org/10.3934/mbe.2019113

[13] Wahab, M. N. A., Lee, C. M., Akbar, M. F., & Hassan, F. H. (2020). Path Planning for Mobile Robot Navigation in Unknown Indoor Environments Using Hybrid PSOFS Algorithm. IEEE Access, 8, 161805–161815. https://doi.org/10.1109/ACCESS.2020.3021605

[14] Li, X., Wu, D., He, J., Bashir, M., & Liping, M. (2020). An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot. Journal of Control Science and Engineering, 2020. https://doi.org/10.1155/2020/3857894