Kufa Journal of Engineering
Vol. 17, No. 2, April 2026, P.P. 669-684
Article history: Received 11 June 2025, last revised 17 August 2025, accepted 23 August 2025

ENHANCEMENT PERFORMANCE OF PATH PLANNING PROCESS BY USING OPTIMAZATION AND INTELLIGENT TRAINING ALGORITHM IN SINGLE AGENT CLOUD ROBOTIC

Hadeel Khudhur Al-Ahmedy[1], Saif Alabachi[2], and Ibrahim Adel Ibrahim[3]

[1] Department of Computer Engineering, University of Technology-Iraq, Baghdad, Iraq, Email:120038@uotechnology.edu.iq.

[2] Department of Computer Engineering, University of Technology-Iraq, Baghdad, Iraq, Email:saif.g.mohammed@uotechnology.edu.iq.

[3] Department of Computer Engineering, University of Technology-Iraq, Baghdad, Iraq, Email:ibrahim.a.almotairi@uotechnology.edu.iq.

https://doi.org/10.30572/2018/KJE/170240

ABSTRACT

The autonomous navigation of robots could be carried out by path planning that enables these robots to move efficiently and avoid obstacles even in complex areas. This work deployed metrics such as task completion time and path length to enhance the performance of path planning. Surpassing the limitations in the traditional approaches, this study improved the decision making in hard environments and ensured safe interaction between human and robots. In addition, this paper used cloud computing to increase the speed of the tasks and work easily in real time with a chance of scalability in the future. The main challenge in the path planning framework is to optimize both time and distance in single agent systems. Therefore, this challenge could be solved by presenting a hybrid path planning method that combined tow algorithms that are Particle Swarm Optimization (PSO) and Elman Neural Network (ENN). Also, this work adopted Message Queuing Telemetry Transport (MQTT) as a lightweight communication protocol to efficient exchange data between robot and cloud. The results showed that the proposed method achieved shortest completion times 5.42 seconds for case study 1, 6.82 seconds for case study 2 and a shortest path of 592.9 centimeter for case study 3. This research contributed to more efficient and reliable robotic path planning that offering a robust solution for navigating complex and dynamic environments.

KEYWORDS

Autonomous mobile robots (AMRs), Elman neural network (ENN), Particle swarm optimization (PSO), Path planning, MQTT protocol, Single-agent robot.

1. INTRODUCTION

Path planning is a critical challenge in robotics; ensuring autonomous agents navigate efficiently while avoiding obstacles. For a single-agent robot, path planning must balance optimality, computational efficiency, and adaptability to dynamic environments. Traditional approaches, such as graph-based algorithms and heuristic search methods, often struggle with real-time constraints and complex environments (Katona et al., 2024). In addition, these methods have a number of shortcomings in situations with intricate obstacles, including high processing requirements. The workload for mobile robots can be greatly increased by this misalignment with real-world conditions (Zhang et al., 2022). Therefore, heuristic methods such as Particle Swarm Optimization (PSO) are widely adopted by researchers to enhance path planning performance, yielding promising results (Wen et al., 2021). Heuristic path planning methods are increasingly important because they emulate human-like learning and decision-making (Abed, Lutfy and Al-Doori, 2021). The PSO algorithm is a widely applied heuristic technique for robotic path planning optimization due to its simple structure and ease of implementation (Fan, Zhang and Li, 2021). However, standard PSO struggles in dynamic or noisy environments and lacks inherent mechanisms for constrained optimization. It also faces challenges in maintaining a proper balance between exploration and exploitation which can lead to either slow convergence or local trapping. Furthermore, without careful tuning, the swarm may stagnate the algorithm's reliability in complex scenarios (Osaba and Yang, 2021). Researchers address the limitations of Particle Swarm Optimization (PSO) by hybridizing it with other optimization techniques. For example, a study (Abaas and Shabeeb, 2022) used PSO that achieved good results in mobile robot navigation, though the hybrid IPSOGWO to further enhanced its performance. The Elman Neural Network (ENN) is particularly effective for modeling dynamic systems and capturing temporal relationships in time-series data. However, its performance is highly dependent on a proper initializations of weights and thresholds. To enhance the effectiveness of ENN, this work (Yang, 2024) introduces an Optimized Particle Swarm Optimization (OPSO) approach that adjusted these parameters before training.

This study (Ab Aziz et al., 2021) introduced a PSO-ERNN metaheuristic model that combines Particle Swarm Optimization (PSO) with an improved Elman Recurrent Neural Network (ERNN). The model is evaluated against the standard PSO and similar models such as breast cancer and heart disease. Results showed that PSO-ERNN performed higher accuracy with larger datasets in most cases. A study (Kamil, Mohamed and Oleiwi, 2020) utilized Artificial Bee Colony (ABC) that is considered part of Artificial Intelligence (AI). ABC is an AI-based optimization algorithm because it uses adaptive, heuristic decision-making (modeled on

honeybee foraging) to search for solutions in problems like path planning, scheduling, or parameter tuning, without needing explicit mathematical formulas for the solution. ADL (Adaptive Dimension Limit) ABC achieved shorter paths and reduced computation time with combing with ADL (Adaptive Dimension Limit). With the advancement of cloud computing and communication technologies, a significant portion of computational tasks can be transferred to the cloud to reduce the processing burden on robots (Rahamathunnisa et al., 2023). By leveraging cloud-based algorithms and artificial intelligence, a single-agent robot can dynamically allocate tasks based on priority, environmental conditions, and available resources (Fadhil, Abed and Jasim, 2021). This work proposed an approach that combined two methods (i.e., Particle Swarm Optimization (PSO) and Elman Neural Network (ENN)) with using cloud computing capability.

2. METHODOLOGY

Particle Swarm Optimization (PSO) works by repeatedly updating each particle's position and velocity until the desired target is reached. The ENN presents rapid convergence that is utilized to improve PSO performance through training on the dataset and offloading computationally intensive processing to the cloud for greater efficiency. This approach described the integration of PSO with ENN within a cloud-assisted single-robot system, aiming to enhance the robot's navigation capabilities and optimize of control strategies.

2.1. Experimental Setup

The experiments were conducted using Spyder (Scientific Python Development Environment) version 5.5.1 within Anaconda Navigator, running Python 3.12.4 on a 64-bit system. The hardware platform was equipped with Intel processor with 8 GB of RAM and 2.3 GHz. This section introduces a block diagram, pseudocode and flowchart of the proposed algorithm.

2.2. Block Diagram of the Process

The proposed approach is explained in details in the following block diagram that divided the concept of the approach into three phases. The first phase gives a brief about initialization parameters, while the second phase is about PSO process and the third phase introduces the neural network process.

2.2.1. Phase 1. Initialization

This stage contains three steps which outlined as follows:

Initializing start and end points: the starting point denoted by (S) and the ending point denoted by (E) in the workspace. In the coordinates space, the both point represented by (x, y).

Obstacles generating: these obstacles are generated randomly with a program as rectangles with changing length and width.

Voronoi Diagram: the graph generating by this diagram supports a roadmap to avoid collision and guide the PSO particles.

2.2.2. Phase 2. Processing of PSO

Initializing PSO: the PSO process started by initializing some particles where each one could be a possible solution. In this step several parameters are specified such as initial position of the particles, search domain, velocity of the particles and inertia weight. Table 1 presents these parameters.

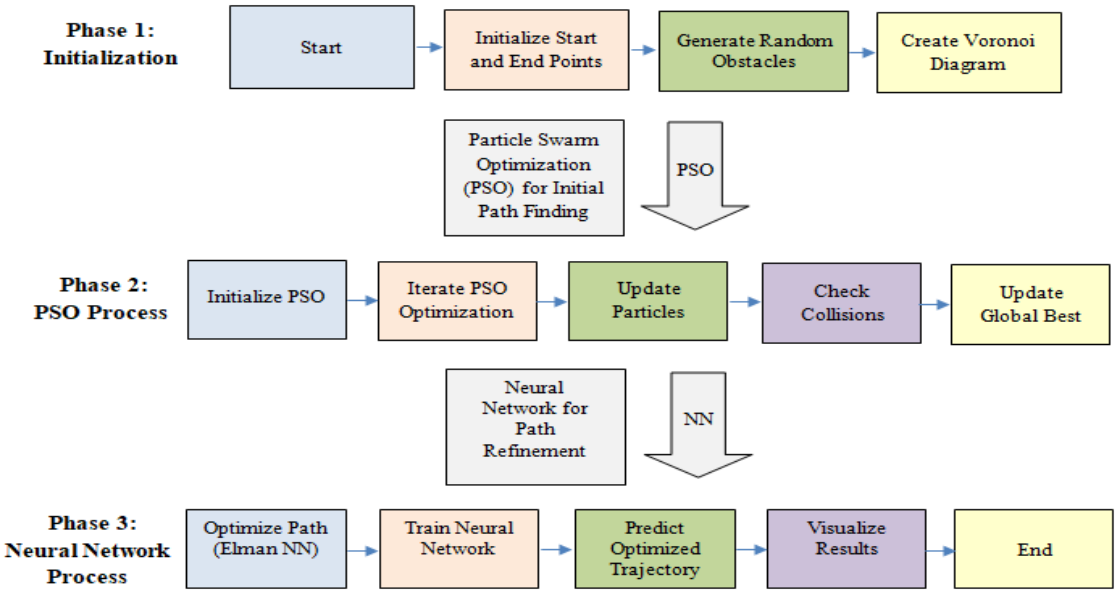

Fig.1.Block Diagram of Hybrid Algorithm (PSO-ENN)

Table1. Initializing parameters of the PSO Algorithm

Number of Iterations	Particles number	Obstacles number	C1	C2	W
100	30	18	0.8	0.9	0.5

Iteration: in this step, the position and velocity of the particles are updating repeatedly in the workspace to find the best solution.

Check Collisions: In the Particle Swarm Optimization (PSO) method, particles explore the search space to find the best solution by repeatedly adjusting their positions and velocities using both their own experience and information shared by neighboring particles.

Global best updating: In Particle Swarm Optimization (PSO), the global best commonly referred to as gbest, represents the highest-quality solution discovered by the swarm. After each iteration, every particle assesses its current fitness and this value is compared with the best solution found previously to update the global best if necessary.

2.2.3. Phase 3: Neural Network Process

Path optimization: the purpose of using PSO is to tune the parameters of the ENN to obtain effective decision for the path of the particles in the space.

Training of neural network: the neural network model trains based on historical data that are produced by varying of obstacle layouts.

Optimal path prediction: to reduce errors in the network or loss function, the optimization process is applied to obtain the best path for the particles. The searching starting by exploring the space and updating large position for the particles.

Visualization the results: the results are presented that obtained from the proposed method. The workspace is set for [20*20] cm with 18 rectangular obstacles with different sizes.

2.3. The Pseudocode for PSO Algorithm

The pseudocode explains the process of the PSO algorithm as shown in Fig. 2.

Step 1: Initialize the swarm with N particle

Step 2: For each particle:
- Initialize position Xi randomly within the search space
- Initialize velocity Vi randomly
- Evaluate the fitness of each particle at position Xi
- Set personal best position Pi = Xi

Step 3: Set global best position as the best Pi among all particles

Step 4: While stopping condition is not met (e.g., max iterations or convergence):
- For each particle:

- Update velocity Vi using the formula:

$$V_i^{t+1} = w\,V_i^t + c_1 r_1 (P_i^t - x_i^t) + c_2 r_2 (G^t - X_i^t)$$

Where:
w is the inertia weight
c1, c2 are acceleration coefficients
r1, r2 are random numbers in [0, 1]
Pi: Personal best position of particle I
G: Global best position among all particles

- Update position : $X_i^{t+1} = X_i^t + V_i^{t+1}, i = 1, \ldots, N$
- Evaluate the new fitness of particle
- Update personal best Pi if new fitness is better
- Update global best if a new best Pi is found

Step 5: Return global best as the optimal solution

Fig.2. The pseudocode for PSO algorithm

2.4. Hybrid Algorithm Flowchart

The sequence of the process for the proposed approach is illustrated in Fig. 3.

Fig.3. The proposed hybrid algorithm flowchart

3. RESULTS

3.1. Simulation Results in Suggested Environments

This section presented the results that were obtained from applying the proposed approach for six different environments with changing the position and shape of the obstacles and as follows:

3.1.1. Environment (1):

The propose approach is carried out in this environment and the results presented in Fig. 4. Table 2 shows the results in details.

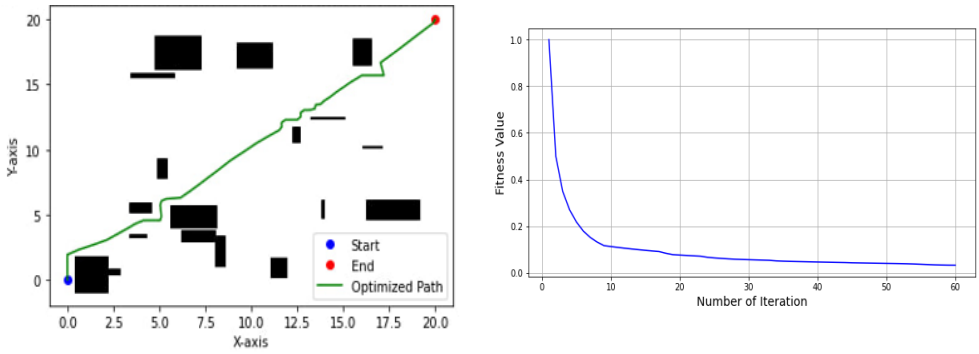

Fig.4. the results for Environment 1: Path Planning (b) Fitness Value

Table2. Parameters of the proposed approach in environment-1

Performance parameters of the proposed approach							
Time (in seconds)			Distance (in centimetre)				
5.97			30.58				
More parameters							
Energy consumption	Computational load	Obstacle clearance	Smoothness	Iteration	Accuracy	scalability	Overall fitness
31.30	1188	0.82	0.12	60	100%	1.66	0.28

3.1.2. Environment (2):

The propose approach is carried out in this environment and the results presented in Fig.5. Table3 shows the results in details.

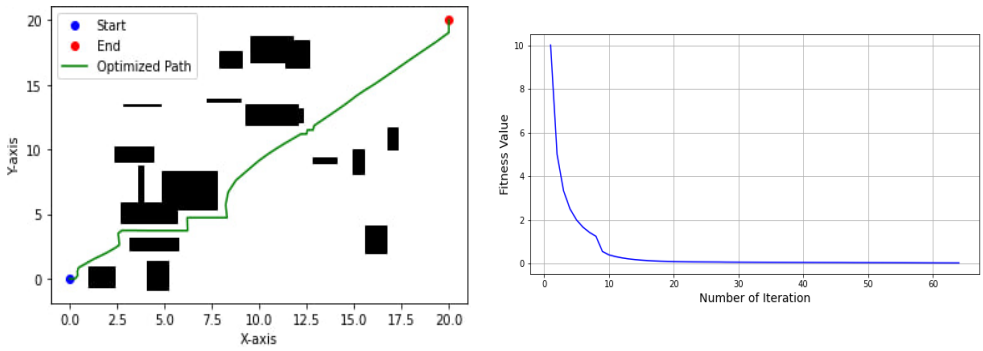

Fig.5. the results for Environment 1:Path Planning (b) Fitness Value

Table3. Configuration Parameters for the Proposed Method (Environment 2)

Performance Metrics of the Proposed Algorithm							
Time (in seconds)					Distance (in centimetre)		
4.86					32.43		
More Metrics							
Computational load	Energy consumption	Obstacle clearance	Smoothness	Iteration	Accuracy	scalability	Overall fitness
1170	32.03	1.32	0.11	66	100%	1.66	0.27

3.1.3. Environment (3):

For environment 3, Fig. 6(a) shows the optimized path, while Fig. 6(b) presents the best fitness value, which was obtained at the 40th iteration out of a total of 55 iterations. The results of evaluation metrics and additional parameters for this environment used the proposed method shown in the Table 4.

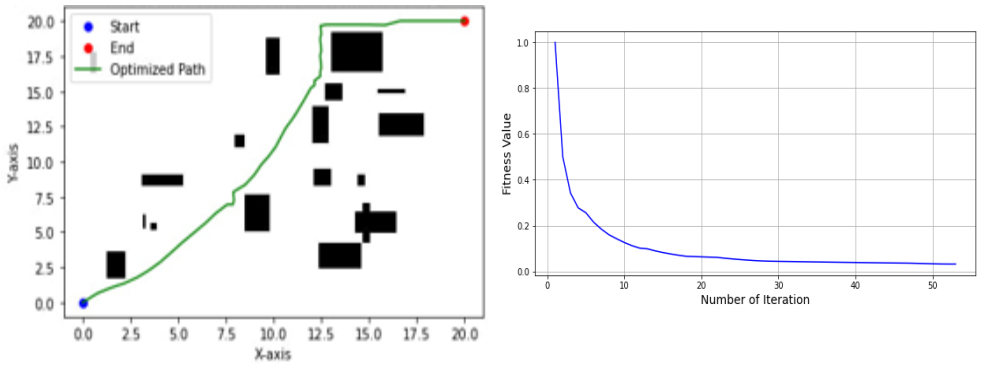

Fig.6. the Proposed Algorithm for Environment-3: Path Planning (b) Fitness Value

Table4. Parameters of the proposed approach in environment-3

Parameters of the proposed approach							
Time (in seconds)				Distance (in centimetre)			
5.28				31.35			
More parameters							
Energy consumption	Computational load	Obstacle clearance	Smoothness	Iteration	Accuracy	Scalability	Overall fitness
32.38	1080	0.88	0.08	55	100%	1.66	0.27

3.1.4. Environment (4):

To demonstrate the proposed method, a different set of obstacles was designed. In Fig. 7(a) presents the path planning, Fig. 7(b) shows the fitness function. The best value was achieved when the iteration reached 65 out of 88 iterations. Moreover, the evaluation measures and supplementary metrics were also calculated for this environment, as reported in Table 5.

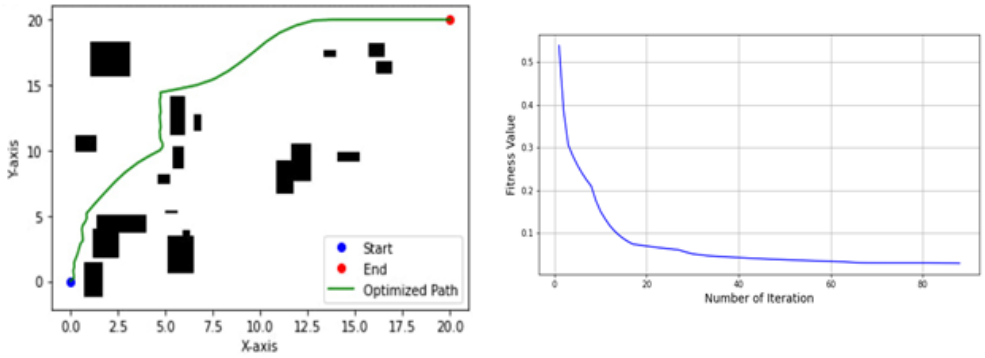

Fig.7. the proposed approach in environment- 4:Path Planning (b) Fitness Value

Table5. Parameters of the Proposed approach in environment-4

Parameters of the Proposed Approach							
Time (in seconds)				Distance (in centimetre)			
5.41				33			
More Parameters							
Energy consumption	Computational load	Obstacle clearance	Smoothness	Iteration	Accuracy	Scalability	Overall fitness
33.32	1026	0.97	0.08	88	100%	1.66	0.27

3.1.5. Two Environments (5) and (6):

In addition, further environments were tested to determine the required time and path length under different obstacle configurations. The optimized paths are shown in Fig. 8(a) and

Fig.9(a). The best fitness value was achieved when iteration reached 45 out of 52 in environment-5 and when iteration reached 30 out of 38 in environment-6, as shown in Fig. 8(b) and Fig. 9(b), respectively. The results for these environments are introduced in the Table 6 and Table 7. Fig.9 and Table 7 for environment (6) show below.

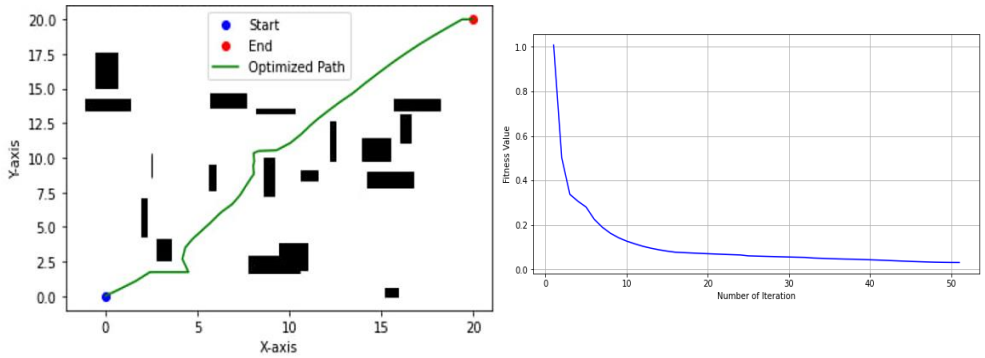

Fig.8. the Proposed Approach in Environment-5:Path Planning (b) Fitness Value

Table 6. Parameters of the Proposed approach in environment-5

Parameters of the Proposed Approach							
Time (in seconds)				Distance (in centimetre)			
5.51				31.43			
More Metrics							
Energy consumption	Computational load	Obstacle clearance	Smoothness	Iteration	Accuracy	Scalability	Overall fitness
30.75	828	1.47	0.11	52	100%	1.66	0.24

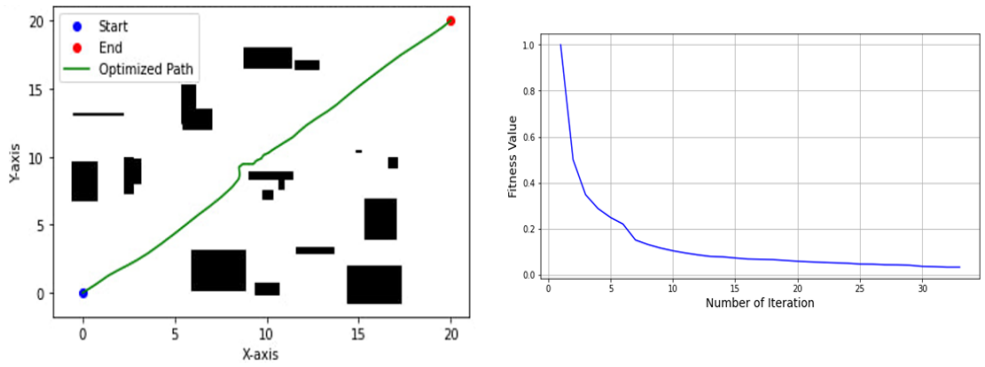

Fig.9. the Proposed Approach in environment-6:Path Planning (b) Fitness Value

Table7. Configuration Parameters for the Proposed Method (Environment 6)

Parameters of the Proposed Approach							
Time (in seconds)				Distance (in centimetre)			
5.08				30.33			
More Metrics							
Energy consumption	Computational load	Obstacle clearance	Smoothness	Iteration	Accuracy	scalability	Overall fitness
28.82	792	2.55	0.10	38	100%	1.66	0.21

3.2. Comparison Results between the Proposed Approach and other Approaches

To prove the effectiveness of the proposed approach in analyzing and solving problems that are related to a robot path planning, it is evaluted by comparing this approach with other

approaches. all simulation experiments were conducted on the same computer using the same Python version. The selected obstacle-based environments are well suited for assessing optimal path planning in a single-robot system. The algorithm's capability was tested with varying levels of complexity, obstacle density, and layout. These factors are essential for evaluating path quality. In contrast, increasing the number of robots primarily addresses system scalability, which is not the focus of this work.

3.2.1. Case Study-1:

The proposed approach is compared with DE, PSO, ABC, and IPSO-IDE to evaluate its performance (Yuan, Sun and Du, 2022). This work (Yuan, Sun and Du, 2022) and the comparison algorithms were evaluated in an environment containing ten obstacles with different shapes. The positions of the start and end points are represented by a square and a symbol "X," respectively, and the range of x and y coordinates are between [0,10]. Fig. 10(a) presents the optimal trajectories obtained by the different methods, while Fig. 10(b) shows the convergence curves of the best fitness values produced by these approaches. In this case, the authors presented a work of a paper as introduced (Yuan, Sun and Du, 2022) in Fig. 11.

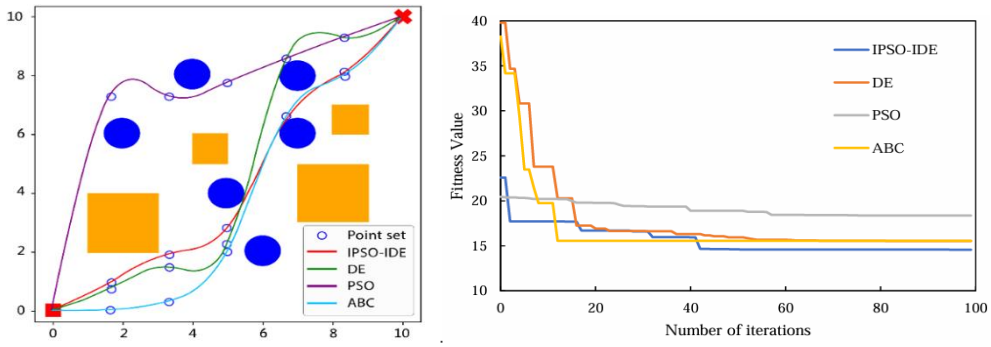

Fig.10. Other Algorithms Based on Previous Algorithms Carried out by (Yuan, Sun and Du,2022) (a) Optimal Path (b) Fitness Values

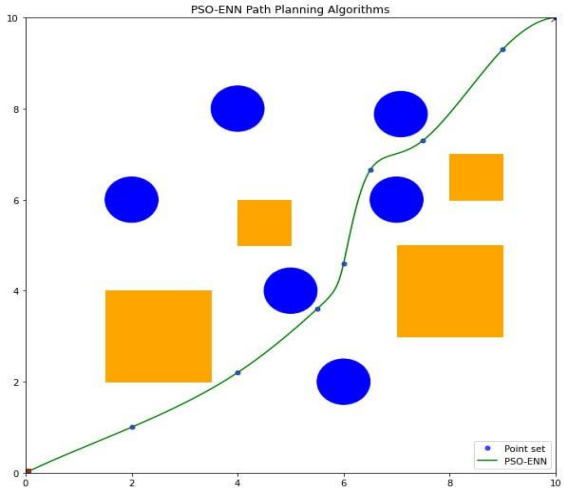

Fig.11. PSO–ENN Algorithm Applied to Case Study 1

Table 8 shows that the proposed method achieved minimum time about 5.42 seconds with 5.83

seconds for the IPSO_IDE. In contrast PSO took 6.53 second to accomplish task and 7.52 seconds for DE.

Table 8. The results for (Case Study 1)

Algorithm	PSO-ENN	IPSO-IDE	DE	PSO	ABC
Time	5.42	5.83	7.52	6.53	7.90

3.2.2. Case Study (2):

In this case, several methods are curried out which are IPSO-IDE, PSO-ABC, DPGPSO, PSO-DE, and IDE to compare results with proposed approach. Fig. 12 presented the results for these algorithms for 100 iterations. In Fig. 13 shows that the PSO–ENN algorithm obtained higher performance compared with the methods IPSO-IDE, PSO-ABC, DPGPSO, PSO-DE, and IDE.

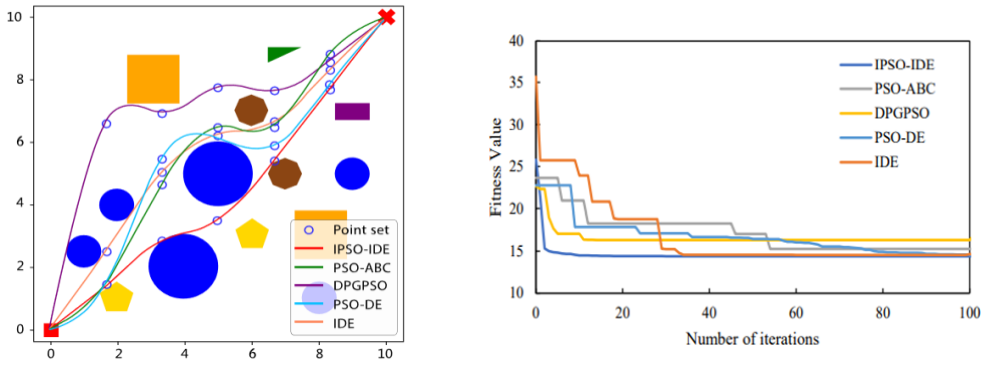

Fig.12. Algorithms suggested by (Yuan, Sun and Du, 2022) (a) Path Planning (b) Fitness Values

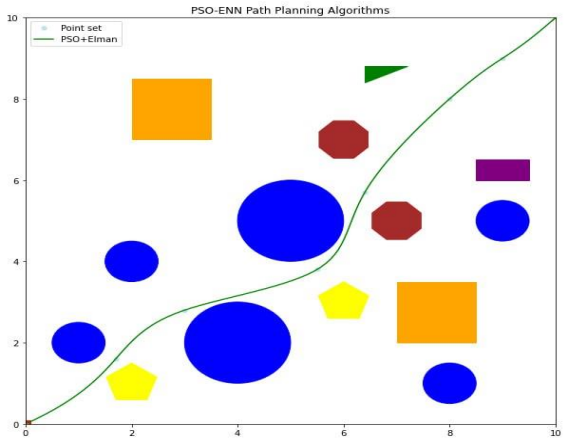

Fig.13. Results of Case study-2 for different methods

Table 9 shows the time in seconds for these algorithms that were suggested by another study. The proposed method presents improvement in the execution time.

Table 9. Execution time for (Case Study 2)

Algorithm	PSO-ENN	IPSO-IDE	PSO-ABC	DPG-PSO	PSO-DE	IDE
Time (s)	6.82	7.08	9.13	7.52	11.13	9.74

3.2.3. Case Study (3):

This case is conducted to assess the performance of the proposed method in comparison with

the hybrid RFD–PSO algorithm (Hassen, Amin and Al-Araji, 2023). The experiment was carried out using the same hardware platform and parameter settings. The proposed method was applied to the same environment previously used for the hybrid RFD–PSO algorithm, which contains 17 obstacles of different shapes and sizes. The robot begins its movement from point (80, 450) cm to point (600, 100) cm. The Fig. 14(a) shows the optimal path, while Fig.14(b) presents the convergence curves of the best fitness values.

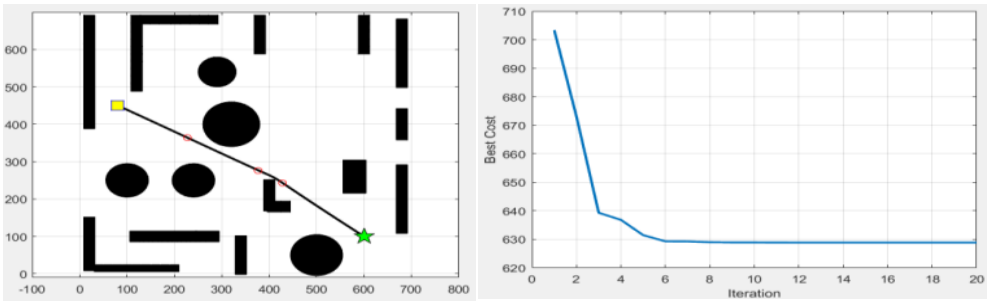

Fig.14. Hybrid RFD-PSO Algorithm Carried out (Hassen, Amin and Al-Araji, 2023) Path planning (b) Fitness Values

The Fig. 15 shows that the proposed approach was implemented to compute the optimal path within the specified workspace. The length of the path that obtained by utilizing the proposed approach is about 593 cm, whereas the (RFD–PSO) algorithm produces a path length of approximately 596.8 cm.

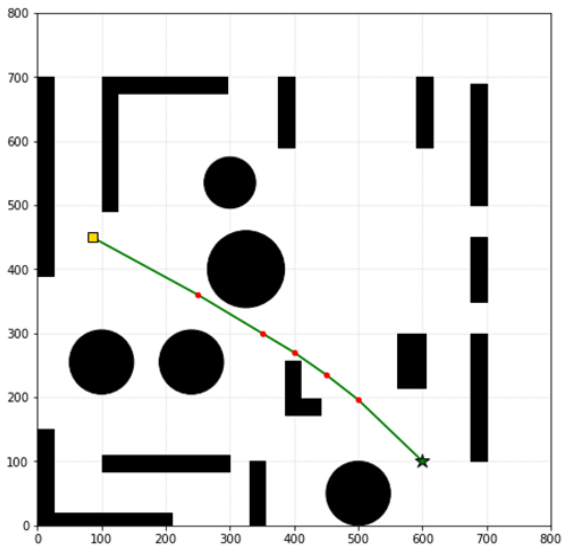

Fig.15. the PSO–ENN algorithm was applied to Case Study 3

As shown in Table 10, the proposed method surpassed the hybrid RFD–PSO algorithm by providing a shorter and more efficient path to the destination.

Table 10. Comparison of the Proposed Algorithm with the Hybrid RFD–PSO Method

Algorithm	Iteration	Distance (cm)
The proposed method (PSO-ENN)	27	592.9
RFD-PSO	20	596.8

4. DISCUSSION THE RESULTS

Regarding the results presented in Section 3.1, the environment-6 has a path that is relatively direct with minimum nodes a fewer numbers of iterations. As the shape of the path in environment-6 is strait forward, it results in the shortest travel distance compared with the other environments. In contrast, Environment 4 exhibits the longest path distance (33 cm), as the robot follows a longer route to avoid obstacles, as illustrated in Fig. 7(a). Despite this, the completion time remained reasonable due to the relatively small number of nodes, which is attributed to effective path smoothing. Environments 3 and 4 showed minimal values for path smoothing and obstacle clearance. The scalability and accuracy parameters remained constant across all environments as the proposed approach relies on cloud computing.

Section 3.2 shows that the proposed algorithm achieved higher results from the other algorithms in three cases. The PSO-ENN obtained a shortest time (5.42 s) in case 1 from the methods PSO, DE, ABC, and IPSO-IDE. For case 2, the proposed algorithm gained shortest time (6.82 s) from the algorithms PSO-ABC, DPG-PSO, and PSO-DE. Also, in case 3, the proposed algorithm achieved a minimum time (592.9 s) from the RFD-PSO method.

5. CONCLUSION

According to the scholars' review that stated the single agent robot with assisting of cloud computing could achieve high performance in making decision in the path planning in less time and short distance. Therefore, this work mixed two algorithms (i.e., PSO and ENN) with utilizing cloud computing. The first algorithm can search the optimal paths efficiently while the second algorithm adapts dynamically in the environments through learning temporal patterns. In addition, cloud computing is utilized to improve the performance of the proposed approach through offloading heavy computations and reducing the robot's on-board processing burden. The results showed that the proposed system success in avoiding the obstacles in 100% through the six different environments and achieving lower fitness values which indicating more efficient and smoother paths. Also, in the comparative analysis, the hybrid model confirmed the superiority across three case studies over traditional algorithms. The proposed approach exhibited excellent application ability, rapid convergence rate and handling of dynamic obstacles. This confirms the method’s potential to be applied in a real-world deployment with autonomous mobile robots.

For future work, multi-agent coordination and hardware implementation will be investigated. Furthermore, the model will be optimized for path planning in a single-agent robot by leveraging more cloud computational power. Overall, the PSO-ENN-cloud framework is a strong and scalable method for intelligent path planning.

ACKNOWLEDGMENTS

The authors would like to thank editor in chief, editors and reviewers of Kufa Journal of Engineering.

CONFLICTS OF INTEREST

The authors have no conflict of relevant interest to this article.

6. REFERENCES

Ab Aziz, M. F., Mostafa, S. A., Mohd. Foozy, C. F., Mohammed, M. A., Elhoseny, M. and Abualkishik, A. Z., (2021). "Integrating Elman recurrent neural network with particle swarm optimization algorithms for an improved hybrid training of multidisciplinary datasets, " Expert Systems with Applications, 183, 115441. https://doi.org/10.1016/j.eswa.2021.115441.

Abaas, T. F. and Shabeeb, A. H., (2022). "Obstacle avoidance and path planning of a wheeled mobile robot using Hybrid algorithm," Engineering and Technology Journal, 40(12), 1–12. doi: 10.30684/etj.2022.132929.1154

Abed, M. S., Lutfy, O. F. and Al-Doori, Q. F., (2021). "A review on path planning algorithms for mobile robots," Engineering and Technology Journal, 39(5A), 804–820. doi: 10.30684/etj.v39i5A.19411

Choudhury, S., Gupta, J. K., Kochenderfer, M. J., Sadigh, D. and Bohg, J., (2022). "Dynamic multi-robot task allocation under uncertainty and temporal constraints," Autonomous Robots, 46(1), 231–247. https://doi.org/10.1007/s10514-021-10022-9

Fadhil, G., Abed, I. and Jasim, R., (2021). "Genetic Algorithm Utilization to Fine Tune the Parameters of PID Controller," Kufa Journal of Engineering, 12(2 SE-Peer-reviewed Articles), 1–12. https://doi.org/10.30572/2018/KJE/120201.

Fan, Q., Zhang, Y. and Li, N., (2021). "An autoselection strategy of multiobjective evolutionary algorithms based on performance indicator and its application," IEEE Transactions on Automation Science and Engineering, 19(3), 2422–2436. doi: 10.1109/TASE.2021.3084741

Hassen, W. M., Amin, S. H. and Al-Araji, A. S., (2023). "Hybrid Swarm Algorithm for Mobile Robot Path Planning," International Journal on Recent and Innovation Trends in Computing and Communication, 11(9), 947-957.

Kamil, R. T., Mohamed, M. J. and Oleiwi, B. K., (2020). "Path planning of mobile robot using improved artificial bee colony algorithm," Engineering and Technology Journal, 38(9), 1384–1395. doi: https://doi.org/10.30684/etj.v38i9A.1100

Katona, K., Neamah, H. A. and Korondi, P., (2024). "Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot," Sensors, 24(11), 3573. https://doi.org/10.3390/s24113573.

Kheder, H. A., (2023). "HUMAN-COMPUTER INTERACTION: ENHANCING USER EXPERIENCE IN INTERACTIVE SYSTEMS," Kufa Journal of Engineering, 14(4 SE-Peer-reviewed Articles), 23–41. https://doi.org/10.30572/2018/KJE/140403.

Osaba, E. and Yang, X.-S., (2021). "Applied optimization and swarm intelligence: a systematic review and prospect opportunities," Applied Optimization and Swarm Intelligence, 1–23. doi: 10.1007/978-981-16-0662-5_1

Rahamathunnisa, U., Sudhakar, K., Murugan, T. K., Thivaharan, S., Rajkumar, M. and Boopathi, S., (2023). "Cloud Computing Principles for Optimizing Robot Task Offloading Processes," In AI-Enabled Social Robotics in Human Care Services (pp. 188–211). IGI Global. doi: 10.4018/978-1-6684-8171-4.ch007

Ramírez-Ochoa, D.-D., Pérez-Domínguez, L. A., Martínez-Gómez, E.-A. and Luviano-Cruz, D., (2022). "PSO, a swarm intelligence-based evolutionary algorithm as a decision-making strategy: A review," Symmetry, 14(3), 455.https://doi.org/10.3390/sym14030455

Wen, J., Yang, J. and Wang, T., (2021). "Path planning for autonomous underwater vehicles under the influence of ocean currents based on a fusion heuristic algorithm," IEEE Transactions on Vehicular Technology, 70(9), 8529–8544. doi: 10.1109/TVT.2021.3097203

Yang, Y., (2024). "An Optimized-PSO Approach for Improving Elman Neural Network Performance in Time-Series Forecasting," 2024 IEEE 2nd International Conference on Electrical, Automation and Computer Engineering (ICEACE), 1712–1718. https://doi.org/10.1109/ICEACE63551.2024.10898678.

Yuan, Q., Sun, R. and Du, X., (2022). "Path planning of mobile robots based on an improved particle swarm optimization algorithm," Processes, 11(1), 26. doi: 10.3390/pr11010026

Zhang, Z., Wu, R., Pan, Y., Wang, Y., Wang, Y., Guan, X., Hao, J., Zhang, J. and Li, G., (2022). "A robust reference path selection method for path planning algorithm," IEEE Robotics and Automation Letters, 7(2), 4837–4844. doi: 10.1109/LRA.2022.3152687.