Al-Khwarizmi Engineering Journal
ISSN (printed): 1818 – 1171, ISSN (online): 2312 – 0789
Vol. 21, No. 2, June, (2025), pp. 42-52

Al-Khwarizmi Engineering Journal

Autonomous Path Planning and Obstacle Avoidance of a Wheeled Mobile Robot via Grey Wolf Optimisation

Zahraa A. Abdulla[1*], and Nabil H Hadi[2]
[1] *Department of Mechanical Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq*
[2] *Department of Aeronautical Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq*
Corresponding Author's Email: zahraa.abdullah2003d@coeng.uobaghdad.edu.iq

(Received 19 May 2024; Revised 2 August 2024; Accepted 1 September 2024; Published 1 March 2025)
https://doi.org/10.22153/kej.2025.09.003

Abstract

The aim of this study was to implement and compare obstacle avoidance for an autonomous wheeled mobile robot (WMR) via the grey wolf optimisation (GWO) algorithm and the artificial bee colony (ABC) algorithm. The study was conducted via three scenarios, each designed to test the performance of the algorithm under different conditions, considering fixed and moving circular obstacles in the surrounding environment. GWO was used to determine the most efficient, shortest and safest path for the WMR from the starting point to the target point. The results showed that the GWO outperformed the ABC. The GWO also enabled the WMR to avoid obstacles faster by 11.8%, 2.8% and 4.6% and with distances shorter by 1.42%, 2.2% and 1.97% for the three scenarios, respectively.

Keywords: *Bees Algorithm; Grey Wolf Optimisation; Wheeled Mobile Robot.*

1. Introduction

Path planning is extensively utilised in autonomous robot navigation to calculate the most effective path between the starting point (SP) and the destination or target point (TP) whilst avoiding obstacles [1,2]. Path planning for optimising path length is an important part of mobile robot navigation. This optimisation challenge involves determining a path that minimises distance whilst conforming to constraints such as collision avoidance in a particular environment. Over time, extensive research has been conducted to overcome this problem, classifying techniques and algorithms as stochastic or deterministic approaches [1]. Linear programming and Newton's method are deterministic algorithms that ignore randomness in their mathematical foundation. By contrast, stochastic approaches are unconstrained by specific mathematical properties; thus, they can attain global optimal solutions across various objective functions. These stochastic algorithms, which are frequently inspired by natural behaviours observed in creatures such as birds, ants, bees and fish, have gained popularity since the 1980s [2]. Numerous algorithms, such as firefly algorithms [3], ant colony optimisation [4], particle swarm optimisation (PSO) [7,8] and grey wolf optimisation (GWO) [9,10], are widely used. GWO is a recently discovered optimisation method used in various relevant disciplines; it closely resembles the hierarchical structure and hunting methods of grey wolf packs, optimising solutions through actions such as tracking, surrounding and pouncing. In contrast to classic algorithms such as PSO and artificial bee colony (ABC), the GWO algorithm is notable for its simplicity; it needs relatively few parameters and follows clear implementation principles. Path planning remains a critical topic in mobile robot research, with the goal of determining collision-free pathways between defined start and target points in a known environment [9]. This problem is further separated into two parts: static path planning, in which obstacles remain fixed, and dynamic path planning, in which obstacle positions change over time. Several techniques and algorithms have been suggested to address the path

planning issue. A relatively new method [10] uses ABC algorithms to plan collision-free trajectories for mobile robots. This methodology reportedly produces the best results and is suitable for real-time applications. The convergence and stochastic stability of the PSO algorithm were analysed [11]. The research was differentiated from traditional PSO by investigating the statistical distributions of the PSO parameters. A comparative analysis [12] was conducted to compare the standard and upgraded versions of the bacterial foraging optimisation algorithm. This investigation resulted in the identification of optimal trajectories. Chaotic sequences [13] were utilised to change parameters and improve optimisation capabilities. Enhanced Q-learning algorithms [14] were introduced to expedite convergence in difficult situations. These algorithms incorporate plant pollination techniques for initialising Q values and optimising path planning via a variable called the 'limit'. Finally, a comparative analysis [15] was implemented to evaluate the effectiveness of chaotic PSO in comparison with typical PSO algorithms in the task of calculating minimal distances whilst navigating around obstacles in static environments. The results demonstrated that chaotic PSO was more efficient than normal PSO in generating shorter paths and achieving faster convergence. This study utilised the classical ABC and GWO algorithms to compare their achievements in terms of identifying the shortest path and achieving the fastest time to reach the target point. The results demonstrated that the GWO algorithm had higher efficiency.

2. Problem formulation

One of the fundamental phases in path planning is environment modelling. The wheeled mobile robot (WMR) is represented as a particle with a radius (r) in a 2D cartesian space (x, y) with static barriers in the environment, each with a radius (R) (Figure 1). The point (0,0) in the bottom left corner of the environment symbolises the robot's SP. The primary goal of applying obstacle avoidance algorithms in local path planning is to generate an online optimal path between the SP and TP without colliding with any obstacles in the surroundings. Three assumptions are considered in this research. In the first assumption, obstacles are depicted as round shapes. In the second assumption, all obstacles are increased by the robot's radius to ensure safe navigation when moving from the SP to the TP; however, in the third assumption, the type of WMR is nonholonomic.

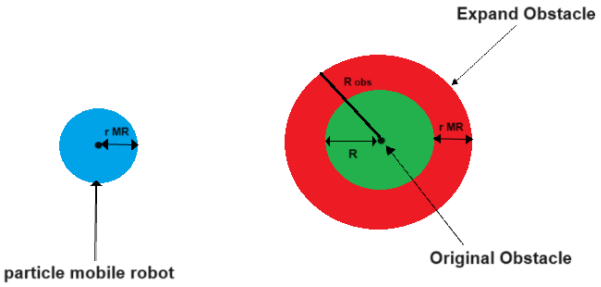


Fig. 1. Original and expanded obstacle boundaries.

3. Performance criteria

3.1. Short path

The shortest path is achieved in planning paths by minimising the distance between the SP and the TP. The mathematical expression used to calculate the function's objective is as follows:

$$F_1 = d\big(p_i(t), p(N)\big) \quad \ldots(1)$$

where d denotes the Euclidean distance, and $p_i(t)$ refers to a set of midpoints the optimisation algorithm selects on the basis of a specified criterion. This criterion requires that these points have the shortest distance to the TP, denoted by $p(N)$. The shortest path distance (SPD) can be determined in the following manner.

$$SPD = \sum_{i=1}^{N-1} \sqrt{(Xp_i(t+1) - Xp_i(t))^2 + (Yp_i(t+1) - Yp_i(t))^2} \quad \ldots(2)$$

where (t) is the simulation time.

3.2. Path smoothness

The goal of obtaining path smoothness is to reduce the angular difference between the present and proposed locations (Figure 2). The mathematical formula used to illustrate path smoothness is given by equations (3) to (5) [16].

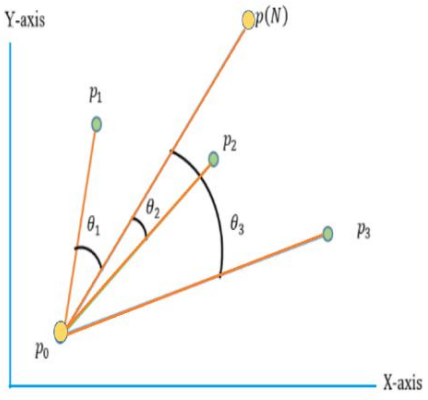


Fig. 2. Example of path smoothness

$$F2 = \sum_{i=1}^{N-1} |\theta(p_i(t), p_i(t+1))\theta(p_i(t), p(N))| \quad ...(3)$$

$$\theta(p_i(t), p_i(t+1)) = \tan^{-1} \frac{Yp_i(t+1) - Yp_i(t)}{Xp_i(t+1) - Xp_i(t)} \quad ...(4)$$

$$\theta(p_i(t), p(N)) = \tan^{-1} \frac{Yp(N) - Yp_i(t)}{Xp(N) - Xp_i(t)} \quad ...(5)$$

The rating of fitness is calculated by combining two objectives, one of which is the weight:

$$F(X,Y) = w1 \cdot F1(X,Y) + w2 \cdot F2(X,Y) \quad ...(6)$$

The values $w1$ and $w2$ indicate the relative significance of the weights applied to each objective. Their individual values must satisfy the following criteria:

$$w = w1 + w2 \quad ...(7)$$

which can explain the following total fitness function:

$$fitness = \frac{1}{F(X,Y) + e} \quad ...(8)$$

Typically, the factor $'e'$ prevents division by zero and is often set to a small value (e.g. 0.001). Each iteration aims to select an optimal solution by balancing the two performance objectives outlined in equations (1) to (3). Amongst the four candidate locations, p2 is optimal for the current iteration 't', and p3 is optimal for the next iteration '(t + 1)' (Figure 3). However, for the subsequent iteration '(t + 2)', whilst the distances for p1 and p4 may be short, the angles are larger. Thus, p2 strikes a balance between these two objectives. This approach continues iteratively until a globally optimal solution is found.

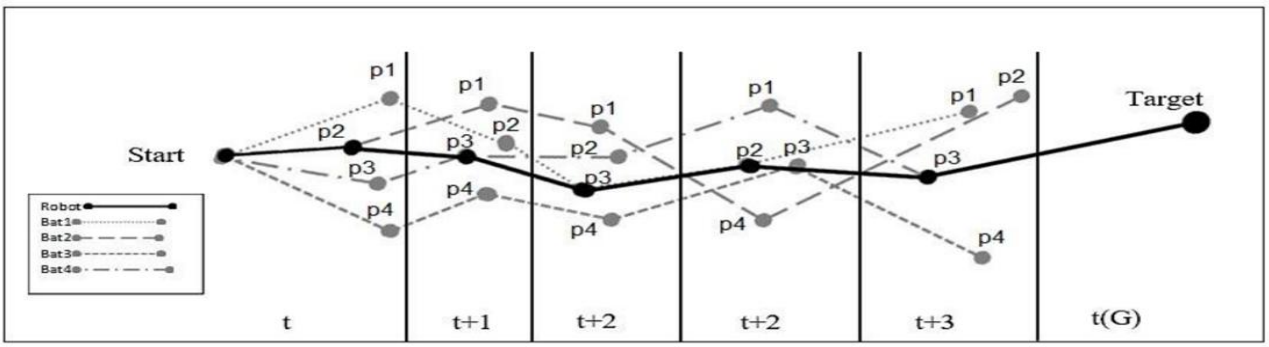


Fig. 3. Planning the path with point selection [16]

3.3. Moving obstacles

Each obstacle might vary its position with each time step. Here, obstacles are assumed to move linearly with constant speed (v_{0bs}) and direction (θ_{obs}) according to the following equations:

$$X_{obs2} = X_{obs1} + v_{0bs} \cdot cos\,\theta_{obs} \cdot t \quad ...(9)$$

$$Y_{obs2} = Y_{obs1} + v_{0bs} \cdot sin\,\theta_{obs} \cdot t \quad ...(10)$$

4. Swarm intelligence

Swarm intelligence (SI) is a branch of study that is based on the collective behaviour of social creatures such as ants, bees, wolves and birds. The goal is to create an artificial algorithm that imitates the characteristics of these natural swarms to address challenging issues. SI algorithms frequently use a population of simple agents or particles interacting with one another and their surroundings to identify optimal solutions. These methods effectively solve optimisation, clustering and routing problems**.** Here, two optimisation methods, namely, GWO and ABC, are utilised to address the path-planning challenge of WMRs.

4.1. ABC algorithm

The ABC algorithm belongs to the SI domain and is influenced by the natural activities of honeybees. Karboga introduced ABC optimisation in 2005 to address complicated situations[17]. According to the ABC framework, bees are classified into three types: employed, scout and onlooker. Employed bees, up to half of the bee population, are principally responsible for finding new food sources and communicating this knowledge to onlooker bees. After receiving information from employed bees, onlooker bees exploit these food sources. A food source depleted by exploitation is marked for replenishment, and a scout bee takes on this responsibility. The basic steps of the ABC algorithm are as follows [19,20]:

4.1.1. Population initialisation

A set of SN individuals is generated randomly. SN indicates the number of solutions, which is equivalent to the number of food sourcesxi = (xi1, xi2, … , xiD). The population searches for positions of food sources. This process continues until the maximum number of cycles is reached. Each solution can be generated according to equation (11) [18].

$x_{i,j} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j})$ ...(11)

where

SN: number of food sources (equivalent to the number of employed or onlooker bees)

D: number of optimisation parameters

j = 1,2, … . . D

i = 1,2 … . SN

$x_{max,j}$ =upper bounds of $x_{i,j}$

$x_{min,j}$ = lower bounds of $x_{i,j}$

rand(0,1) = real number in the interval [0, 1]

4.1.2 Employed bee stage

At this stage, a new solution $v_{i,j} = \{v_{i,1}, v_{i,2}, \ldots . . v_{i,D}\}$ is constructed for food source xi, with only one parameter being updated by

$v_{i,j} = x_{i,j} + \emptyset_{i,j}(x_{i,j} - x_{k,j})$ ...(12)

where $k \in [1,2,3 \ldots \ldots . SN]$ and $j \in, k \neq$i, $\emptyset_{i,j}$ denotes a real number selected randomly from the interval [−1, 1]. Each worker bee subsequently utilised the greedy method to calculate the difference in nectar quantity between $(x_{i,j})$ and $(v_{i,j})$. When the quantity of nectar in $(v_{i,j})$ is greater than that of (xi,j), $(v_{i,j})$ is substituted for $(x_{i,j})$, and the resulting population number is $v_{i,j}$. If $(x_{i,j})$ changes, then counter (i) is incremented by 1.

$$fitness_i = \begin{cases} 1 + abs(f(x_i)), & f(x_i) < 0 \\ \frac{1}{1+f(x_i)}, & f(x_i) \geq 0 \end{cases} \quad ...(13)$$

where $f(x_i)$ is the objective value for the solution x_i calculated via fitness equation (8).

Each onlooker bee's principal function is to determine the food's location by applying the probability (pi) [19]:

$$pi = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \quad ...(14)$$

The $(fitness_i)$ value of the solution x_i is directly proportional to the quantity of nectar present in that particular food source. Once the onlooker bees have selected the food positions, new positions for the food are generated $v_{i,j}$ via equation (12). The bees then employ a similar greedy strategy to assess these new positions.

4.1.3 Scout bee stage

The abandonment counter (count I) of each solution x_i is evaluated in relation to the limit. The algorithm maintains a constant limit throughout its execution. The employed bees, which cannot improve their self-solution through the limit, are considered the worst. The bees are then dispatched as scout bees, in accordance with equation (11), to arbitrarily investigate new food sources after discarding the poorest food source.

4.2 GWO algorithm

According to research, grey wolves, a type of canid animal, play an important role as apex predators in the ecological food chain, indicating that they are top predators. These creatures greatly prefer communal living, which involves a complex social hierarchy amongst their packs (Figure 4). The α wolf is at the top of the hierarchy, functioning as the pack's leader and being in charge of controlling group dynamics and making key decisions. The β wolves are located below the α wolf and are responsible for helping them whilst being prepared to take on leadership positions if necessary. The δ wolves are on the third level, obediently carrying out orders from higher-ranking commanders and doing jobs such as surveillance and reconnaissance. Finally, the omega ω wolves at the bottom level are largely focused on preserving peaceful interactions within the pack [23,[21].

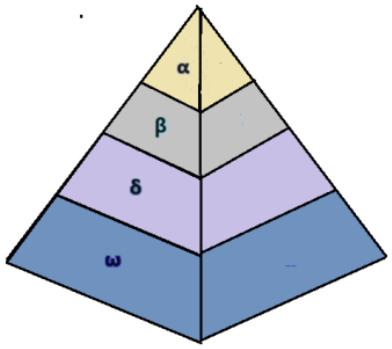


Fig. 4. Grey wolf class scheme

The GWO algorithm completes one cycle in four stages: hierarchical division, exploration, surrounding and attacking. Initially, the wolf population is separated into hierarchies depending on individual fitness levels, symbolised by the letters α, β, δ and ω. The algorithm's optimisation phase concentrates on each generation's top three optimal solutions. The mathematical models that define grey wolves' exploration and tracking

behaviours are outlined below: grey wolves browse the search space to locate probable solutions, simulating their hunting behaviour via a mathematical model. Surrounding: When grey wolves locate a prospective candidate solution or prey, they work together to surround it, producing an encirclement pattern to increase the chances of catching it. Attack: Finally, after successfully surrounding the prey, grey wolves select the most appropriate individual within the surroundings to start an attack, with the goal of increasing the population's overall fitness. Figure 5 shows solid circles representing the positions of wolves labelled α, β, δ and ω in a 2D plane. The symbol P denotes the prey's relative position. Following the three major processes of grey wolf hunting, i.e. approaching, surrounding and attacking the prey, the α wolf, accompanied by β and δ wolves, leads the pursuit once the prey is located. Amongst the wolves, α, β and δ are closest to the prey, and their placements influence the direction of prey P. When the fitness of the grey wolves is assessed, optimal, good and suboptimal options can be identified. The positions of the other wolves are based on those of α, β and δ. Each wolf in the pack indicates a possible option for the population. The location of the α wolf represents the optimal answer, and the positions of β and δ reflect good and inferior solutions. $\overrightarrow{x}$ indicates a grey wolf's current position vector. $\overrightarrow{x_\alpha}$, $\overrightarrow{x_\beta}$ and $\overrightarrow{x_\delta}$ represent the three best individuals. The vectors $\overrightarrow{C_1}$, $\overrightarrow{C_2}$ and $\overrightarrow{C_3}$ are random, and $\overrightarrow{D_\alpha}$ and $\overrightarrow{D_\delta}$ represent the distances between the candidate wolf and the three best individuals. The vectors $\overrightarrow{x_1}$, $\overrightarrow{x_2}$, and $\overrightarrow{x_3}$ represent the candidate wolves' step lengths towards the three best individuals. Figure 5 shows the execution concept for the GWO algorithm.

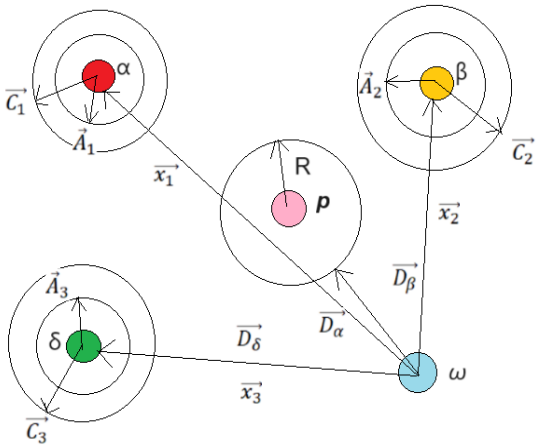


Fig. 5. Schematic of GWO principles

The mathematical framework for how grey wolves look for and track prey is expressed as follows:

$$\vec{D} = \left|\vec{C} \cdot \vec{x}_p(t) - \vec{x}(t)\right| \quad \ldots(15)$$

$$\vec{x}(t+1) = \vec{x}_{p(t)} - \vec{A} \cdot \vec{D} \quad \ldots(16)$$

where $\vec{D}$ represents the vector reflecting the distance between the gray wolf and its prey. The prey position vector is $\vec{x}_p$, the grey wolf position vector is $\vec{x}$, the iteration number is t, and the coefficient vectors are $\vec{A}$ and $\vec{C}$. The coefficient vectors $\vec{A}$ and $\vec{C}$ can be defined as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad \ldots(17)$$

$$\vec{C} = 2\vec{r}_2 \quad \ldots(18)$$

The convergence factor, denoted by $\vec{a}$, decreases gradually from 2 to 0 with each iteration. The values of $\vec{r}_1$ and $\vec{r}_2$ are random numbers ranging from 0 to 1. When grey wolves find their prey whilst searching, they encircle it using α, β and δ as guides. However, because the specific location of the optimal prey in the abstract space is unknown, the top three best solutions identified thus far are preserved to define positions and motivate other individuals to alter their search positions to mimic the true behaviour of grey wolves. The mathematical model for this method is given by the following:

$$\overrightarrow{D_\alpha} = \left|\vec{C}_1 \cdot \vec{x}_\alpha - \vec{x}\right| \quad \ldots(19)$$

$$\overrightarrow{D_\beta} = \left|\vec{C}_2 \cdot \vec{x}_\beta - \vec{x}\right| \quad \ldots(20)$$

$$\overrightarrow{D_\delta} = \left|\vec{C}_3 \cdot \vec{x}_\delta - \vec{x}\right| \quad \ldots(21)$$

$$\overrightarrow{x_1} = \vec{x}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha \quad \ldots(22)$$

$$\overrightarrow{x_2} = \vec{x}_\beta - \vec{A}_2 \cdot \vec{D}_\beta \quad \ldots(23)$$

$$\overrightarrow{x_3} = \vec{x}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \quad \ldots(24)$$

$$x(t+1) = \frac{\overrightarrow{x_1} + \overrightarrow{x_2} + \overrightarrow{x_3}}{3} \quad \ldots(25)$$

5. Suggested Algorithm

This section covers the two primary subjects of the path planning method for nonholonomic WMRs. The subjects under discussion include the development of feasible paths and the detection of obstacles.

5.1. Feasible path generation

At each iteration (t), the ABC and GWO provide multiple solutions. The task is to select the most suitable via-points from the set of possible options. The choice of via-points is contingent upon formulating the objective function specified in equation (6). This process continues until the WMR reaches its target point.

5.2 Procedure for obstacle detection

Sensing is accomplished by attaching 12 virtual sensors around the mobile robot. These sensors have equal distances, with each sensor covering an angle range of 30° and having a certain sensing range (SR) value (Figure 6). In the research design, the SR is 0.4 m.

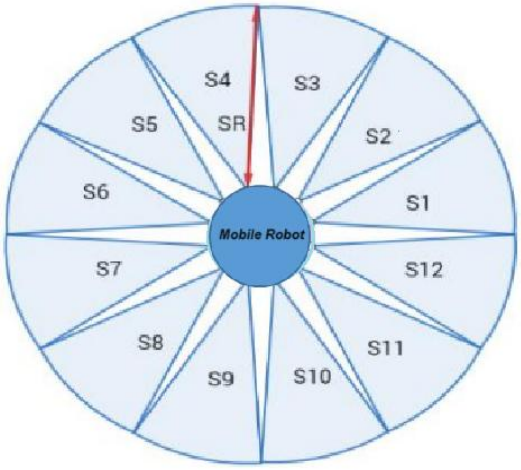


Fig. 6. Mobile robot sensor deployment

The sensory vector (v_s) is employed as a binary vector to indicate whether or not environmental impediments are present. Each of the 12 binary values that comprise SV= [S (1), S (2)..., S (12)] represents a deployed sensor. The equation S(1) =S (3) =S (4) =logic '1' might be used to identify obstacles within the DR range of S1, S3 and S4 and the DA range of S4. Otherwise, logic '0' means to release space in the surrounding surroundings. Here, two environmental impediments are assumed. If the first obstacle is expressed as SV=[1 0 1 1 0 0 0 0 0 0 0 0], then it evidently falls within the immediate scope of S1. However, the second obstacle is located between S3 and S4. This process is completed by determining the geometrical distance between the obstacle and the location of the WMR, and the following formula is used.

$$d(MR_{pos}, Obst_{pos}) \quad \ldots(26)$$

6. Results and Discussion

Three scenarios were tested to evaluate the performance of the implemented algorithms via MATLAB and graphical user interface programming. In the first scenario, six fixed obstacles are present in the surrounding environment. In the second scenario, three moving obstacles are present in the surrounding environment. In the third scenario, five obstacles, three fixed and two moving obstacles, are present. The setting had dimensions of 10 m × 10 m. The parameters consisted of a population size of 0.5 and a WMR radius of 0.15 m, with the SP at (0,0), the TP at (10,10) m, a WMR velocity of 0.2 m/s and a total of 100 iterations. Each scenario was replicated ten times to evaluate the efficacy of the two algorithms.

6.1. First scenario

Six fixed obstacles with three via points are employed. The details of the obstacles are shown in Table 1.

Table 1.
Description of the obstacles in the first scenario

Obstacles	Centre (x, y) m	Radius (unit) m
1	(1,1)	0.2
2	(3,4)	0.25
3	(4,6)	0.4
4	(8,4)	0.3
5	(7,8)	0.2
6	(6,8)	0.4

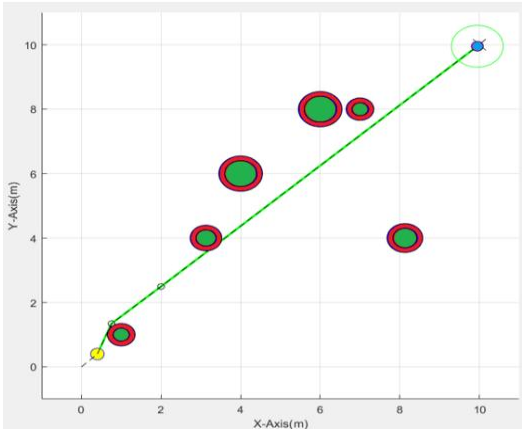


Fig. 7. Optimal path suggested by ABC (fixed obstacles)

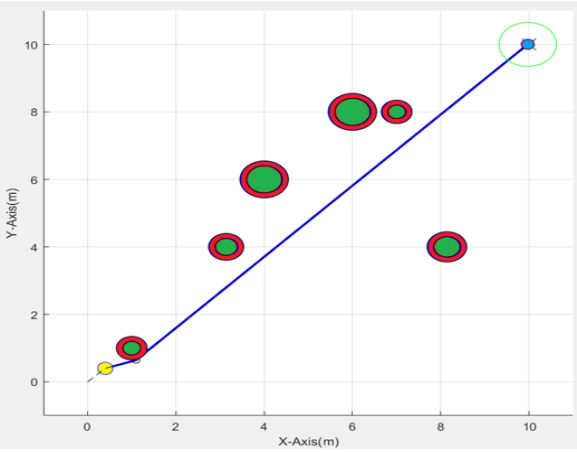


Fig. 8. Optimal path suggested by GWO (fixed obstacles)

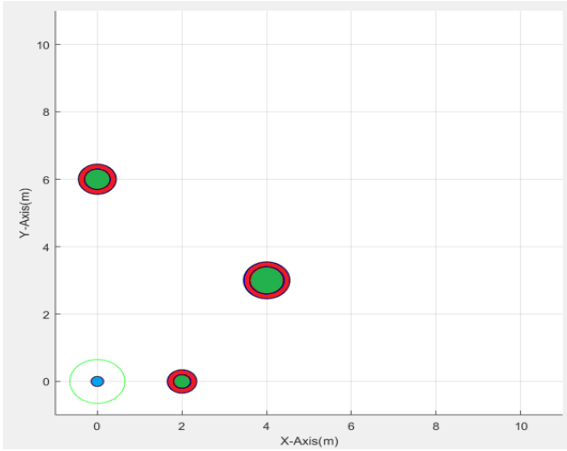
On the basis of the presented plots, specifically Figures 7 and 8, the path produced by the ABC method has a length of 14.255 m. The computation time for generating this path is 27.0246 s. Additionally, the via-points are located at coordinates (2, 2.435), (5.32, 5, 61) and (8, 8.122). By contrast, the GWO algorithm produces a path that has a length of 14.0541 m. The computation time for this approach is 24.0114 s. The more significant via-points are situated at coordinates (2.01, 1.6074), (5.02, 4.7808) and (7.988, 7.71). The analysis of the performance data from Figures 7 and 8 reveal that the path produced by GWO is approximately 1.42% shorter and approximately 11.8% faster than that produced by ABC. Therefore, GWO outperforms ABC because of the improvement in the control limit parameter in GWO.

6.2. Second scenario

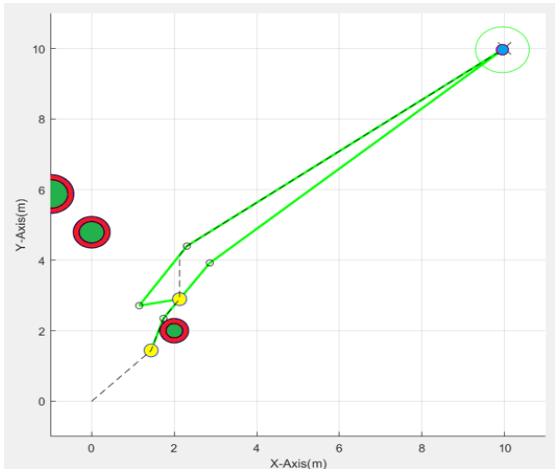
Three moving obstacles with two waypoints are implemented. The obstacle parameters are shown in Table 2.

Table 2.
Description of the obstacles in the second scenario

Obstacles	Initial (x, y) m	Radius (unit) m	$v_{obstacle}$ (m/s)	$\theta_{obatacle}$ (degree)
1	(2,0)	0.2	0.01	90
2	(4,3)	0.4	0.05	150
3	(0,6)	0.3	0.01	270

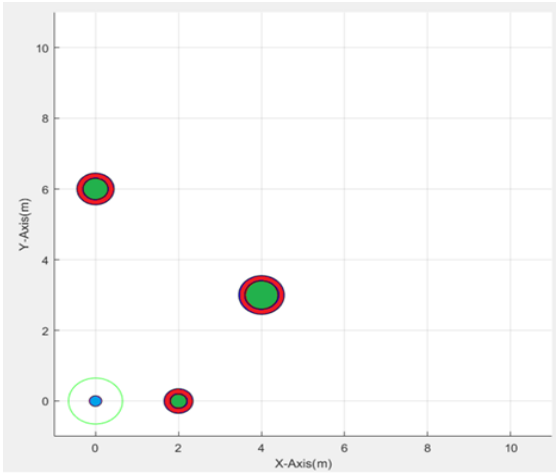


(A)

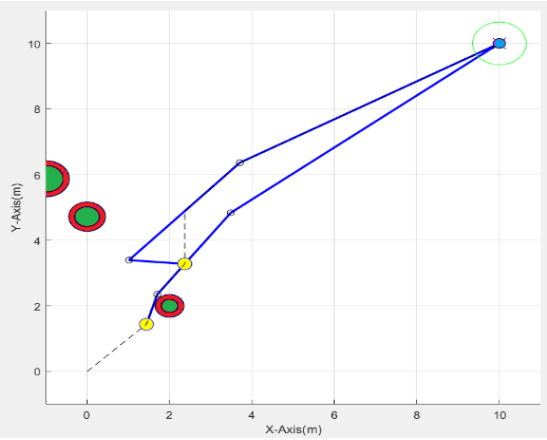


(B)

Fig. 9. Optimal path suggested by ABC (moving obstacles)



(A)



(B)

Fig. 10. Optimal path suggested by ABC (moving obstacles)

The robot starts its path from the SP at (0,0) and moves towards the TP at (10,10) m (Figures 9(A) and 10(A)). Figures 9(B) and 10(B) show where the robot decided to deviate from its initial path and instead used the ABC and GWO algorithms to create new paths to avoid moving obstacles in the environment. The robot securely reaches the goal point (10 m × 10 m). The total length of the path from the SP to the TP in Figure 9 is equal to 14.8913 m, and the computation time is 37.433 s. The positions of the more significant via-points are (3.21, 5.817) and (7.34, 8). The total path length from the SP to the TP in Figure 10 is 14.566 m, and the computation time is 36.4 s. The positions of the more significant via-points are (3.02, 4.917) and (7.07, 7.866). The results showed that the optimised path generated by GWO was 2.2% shorter and 2.8% faster than that generated by ABC. The GWO algorithm performs better than ABC does, and the algorithm's capacity to determine the quickest and shortest optimal path is enhanced.

6.3. Third scenario

Five obstacles are considered. The third and fourth obstacles are moving, and the others are fixed with three via-points. The details of the obstacles are shown in Table 3.

Table 3.
Description of the obstacles in the third scenario

Obstacles	Centre (x, y) m	Radius (unit) m	$v_{obstacle}$ (m/s)	$\theta_{obatacle}$ (degree)
1	(1,1)	0.4	-------	--------
2	(2,3)	0.25	-------	--------
3	(6,3)	0.3	0.05	60
4	(2,5)	0.25	0.01	90
5	(7,7)	0.35	-------	-------

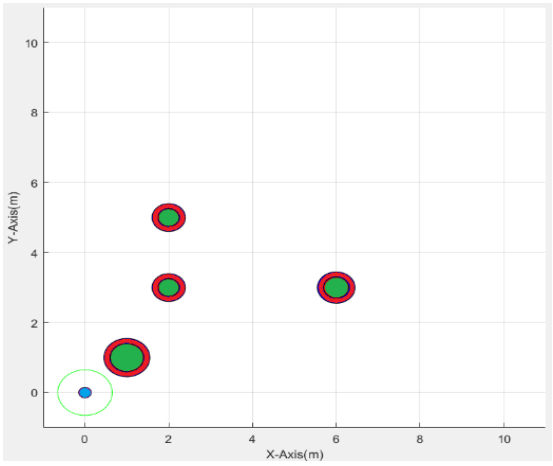


Fig. 11. Robot and obstacle locations for the third scenario before traveling begins

Figure 11 shows the robot's position and the fixed and moving obstacles before it begins moving to take the right path, which allows it to avoid obstacles and safely reach its target point.

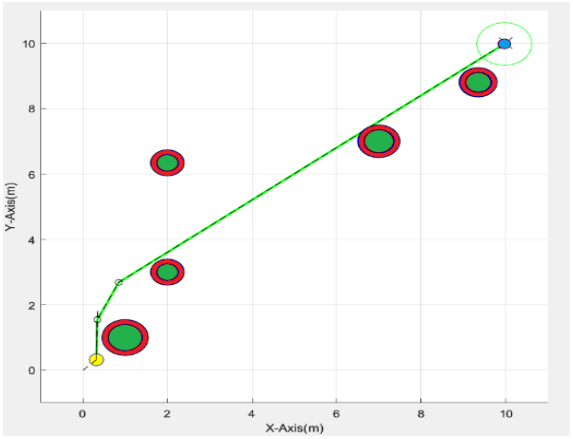


Fig. 12. Optimal path suggested by ABC (fixed and moving obstacles)

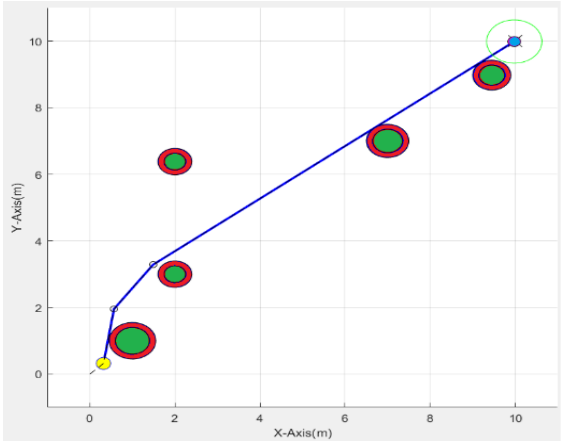


Fig. 13. Optimal path suggested by GWO (fixed and moving obstacles)

Figures 12 and 13 show the paths produced by the ABC and GWO methods. The path of the ABC algorithm has a length of 14.7233 m. The computation time for the robot traveling is 27.8 s. Additionally, the more significant via-points are located at coordinates (1.33, 3.07), (2.53, 4.029) and (7.17, 7.7381). By contrast, the GWO algorithm produces a path with a length of 14.4361 m. The computation time for this approach is 26.548 s. The more significant via-points are situated at coordinates (1.32, 3.13), (2.54, 4.17) and (7.18, 7.7761). The performance data shown in Figures 12 and 13 reveal that the path produced by GWO is approximately 1.97% shorter and approximately 4.6% faster than that produced by ABC. Thus, GWO outperforms ABC because of the improvement in the control limit parameter in GWO.

7. Conclusion

The ABC or GWO algorithms are triggered to create a safe, smooth and faster alternative path to the TP. The GWO algorithm is compared with the classical ABC algorithm in different scenarios to test how sufficiently each algorithm performs, and metrics involving the calculation time and path length are evaluated. The simulation results demonstrated that the ideal path produced by GWO is approximately 1.42% shorter and 11.8% faster than the path generated by ABC in the initial scenario, which includes six fixed environmental obstacles. The second example has three moving obstacles in the environment. The GWO algorithm yields a path that is 2.2% shorter and 2.8% faster than the other paths. The third scenario has five obstacles, consisting of two moving obstacles and three fixed obstacles. Furthermore, the GWO algorithm outperforms the ABC algorithm by following a path that is 1.97% shorter and 4.6% faster than the other paths.

References

[1] K. Heero, *Path planning and learning strategies for mobile robots in dynamic partially unknown environments*. Tartu University Press, 2006.

[2] H. Chen, Y. Zhu, and K. Hu, "Adaptive bacterial foraging optimization. abstract and applied analysis," *Abstr. Appl. Anal. https//doi. org/10.1155/2011/108269*, 2011.

[3] X.-S. Yang, *Optimization techniques and applications with examples*. John Wiley & Sons, 2018.

[4] A. M. Husain, S. M. Sohail, and V. S. Narwane, "Path planning of material handling robot using Ant Colony Optimization (ACO) technique," *Int. J. Eng. Res. Appl.*, vol. 2, no. 5, pp. 1698–1701, 2012.

[5] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Comput.*, vol. 22, pp. 387–408, 2018.

[6] H. Zhangfang, F. Chunyi, and L. Yuan, "Improved particle swarm optimization algorithm for mobile robot path planning," *Comput. Appl. Res*, vol. 38, pp. 3089–3092, 2021.

[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[8] J. Cai, "Non-linear grey wolf optimization algorithm based on Tent mapping and elite Gauss perturbation. Comput," *Eng. Des*, vol. 43, pp. 186–195, 2022.

[9] M. J. Mohamed and F. S. Khoshaba, "Enhanced Genetic Algorithm Based on Node Codes for Mobile Robot Path Planning," *Iraqi J. Comput. Commun. Control Eng.*, vol. 12, no. 2, pp. 69–80, 2012.

[10] E. García-Gonzalo and J. L. Fernández-Martínez, "Convergence and stochastic stability analysis of particle swarm optimization variants with generic parameter distributions," *Appl. Math. Comput.*, vol. 249, pp. 286–302, 2014.

[11] N. H. Abbas and F. M. Ali, "Path planning of an autonomous mobile robot using enhanced bacterial foraging optimization algorithm," *Al-Khwarizmi Eng. J.*, vol. 12, no. 4, pp. 26–35, 2016.

[12] M. Jasim Mohamed and M. Waad Abbas, "Enhanced GA for Mobile Robot Path Planning Based on Links among Distributed Nodes," *Eng. Technol. J.*, vol. 31, no. 1A, pp. 26–41, 2013.

[13] E. S. Low, P. Ong, and K. C. Cheah, "Solving the optimal path planning of a mobile robot using improved Q-learning," *Rob. Auton. Syst.*, vol. 115, pp. 143–161, 2019.

[14] D. Karaboga and B. Gorkemli, "Solving traveling salesman problem by using combinatorial artificial bee colony algorithms," *Int. J. Artif. Intell. Tools*, vol. 28, no. 01, p. 1950004, 2019.

[15] E. A. Hadi and S. By, "Multi-Objective Decision Maker for Single and Multi-Robot Path Planning." M. Sc. Thesis, Control and System Dept., University of Technology,

Baghdad-Iraq, 2018.
[16] B. M. Abed and W. M. Jasim, "Hybrid approach for multi-objective optimization path planning with moving target," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 29, no. 1, pp. 348–357, 2023.
[17] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," 2005.
[18] N. H. Abbas and F. M. Ali, "Path planning of an autonomous mobile robot using directed artificial bee colony algorithm," *Int. J. Comput. Appl.*, vol. 96, no. 11, pp. 11–16, 2014.
[19] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, 2009.
[20] L. Sun, B. Feng, T. Chen, D. Zhao, and Y. Xin, "Equalized grey wolf optimizer with refraction opposite learning," *Comput. Intell. Neurosci.*, vol. 2022, 2022.
[21] J. Li and F. Yang, "Task assignment strategy for multi-robot based on improved Grey Wolf Optimizer," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 12, pp. 6319–6335, 2020.

تخطيط المسار المستقل وتجنب العوائق للروبوت المتحرك ذو العجلات باستخدام تحسين الذئب الرمادي

زهراء عبد[١]*، نبيل حسن هادي[٢]

[١] *قسم الهندسة الميكانيكية، كليةالهندسة، جامعة بغداد، بغداد، العراق*

[٢] *قسم هندسة الطيران، كليةالهندسة، جامعة بغداد، بغداد، العراق*

* البريد الالكتروني: zahraa.abdullah2003d@coeng.uobaghdad.edu.iq

المستخلص

هدفت هذه الدراسة إلى تنفيذ ومقارنة تجنب العوائق لروبوت متحرك ذاتي الحركة (WMR) باستخدام خوارزمية تحسين الذئب الرمادي (GWO) وخوارزمية مستعمرة النحل الاصطناعي (ABC). أجريت الدراسة من خلال ثلاثة سيناريوهات، تم تصميم كل منها لاختبار أداء الخوارزمية في ظل ظروف مختلفة، مع مراعاة العوائق الدائرية الثابتة والمتحركة في البيئة المحيطة. تم استخدام خوارزمية GWO لتحديد المسار الأكثر كفاءة وأقصر وأكثر أمانًا لروبوت WMR من نقطة البداية إلى موقع الهدف. أظهرت النتائج أن خوارزمية GWO تفوقت على خوارزمية ABC، مما مكن WMR من تجنب العوائق بشكل أسرع بنسبة ١١,٨٪ و ٢,٨٪ و ٤,٦٪، وبمسافة أقصر بنسبة ١,٤٢٪ و ٢,٢٪ و ١,٩٧٪ للسيناريوهات الثلاثة على التوالي.