

Multi-objective optimization path planning with moving target

Baraa M. Abed, Wesam M. Jasim

Department of Computer Science, College of Computer Science and Information Technology, University of Anbar, Ramadi, Iraq

Article Info

Article history:

Received Oct 30, 2021

Revised May 15, 2022

Accepted Jun 13, 2022

Keywords:

Bat algorithm

Dynamic environment

Moving target

Multi objective optimization

Particle swarm optimization

Path planning

ABSTRACT

Path planning or finding a collision-free path for mobile robots between starting position and its destination is a critical problem in robotics. This study is concerned with the multi objective optimization path planning problem of autonomous mobile robots with moving targets in dynamic environment, with three objectives considered: path security, length and smoothness. Three modules are presented in the study. The first module is to combine particle swarm optimization algorithm (PSO) with bat algorithm (BA). The purpose of PSO is to optimize two important parameters of BA algorithm to minimize distance and smooth the path. The second module is to convert the generated infeasible points into feasible ones using a new local search algorithm (LS). The third module obstacle detection and avoidance (ODA) algorithm is proposed to complete the path, which is triggered when the mobile robot detects obstacles in its field of vision. ODA algorithm based on simulating human walking in a dark room. Several simulations with varying scenarios are run to test the validity of the proposed solution. The results show that the mobile robots are able to travel clearly and completely safe with short path, and smoothly proving the effectiveness of this method.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Baraa M. Abed

Department of Computer Science, College of Computer Science and Information Technology, University of Anbar

Ramadi, Iraq

Email: burasoft@gmail.com

1. INTRODUCTION

Previously, using robot implemented only in the manufacturing industry. However, nowadays, mobile robots are commonly used in diverse fields such as entertainment, medicine, mining, rescue, education, military, aerospace and agriculture. Smart equipment enables mobile robots to model the environment, determine its location, control movement and discover obstacles while carrying out its tasks. These functions can be performed through navigation technology, of which the most important function is to plan a safe path by detecting and avoiding obstacles when moving from one point to another. Therefore, in planning the robot path for the customer in an environment, whether simple or complex, the most important step is the correct choice of navigation technology [1], [2]. Path planning can be divided into two types based on the robot's knowledge of its environment: global (or offline) and local (or online). Mobile robots have complete knowledge of their environment when it comes to global path planning. Before the robot starts moving, the algorithm generates a complete path for it to follow. Local path planning is performed by mobile robots that have no prior knowledge of their environment and rely on a local sensor to collect data and then construct a new path in response [3]. Depending on the target type: static or dynamic, the above classification can be subdivided. In static targets, the mobile robot looks for a static point in its workspace, but for dynamic targets, it searches for a moving point while avoiding obstacles [4]. The path planning algorithms for those

scenarios are different. Path planning problems in a variety of fields have been solved using various methods such as cell decomposition and roadmap approaches., the main disadvantages of this method include inefficiencies due to high processing fees and unreliability due to significant risks of being caught in local minima. These limitations can be overcome by using different heuristic techniques, such as neuronal pathways, genetics, or nature-inspired algorithms [5]. Several techniques have been used to solve the path planning problem, for a variety of reasons, including safety or single-objective optimisation for the shortest path, these methods such as bacterial foraging optimization (BFO) [6], bat algorithm (BA) [7], cuckoo search (CS) [8], whale optimization algorithm (WOA), particle swarm optimization (PSO) [9], and artificial immune systems (AIS) [10]. Additionally, fuzzy logic and neural networks have been used. Another objective of these solutions is multi-objective optimization, which tries to satisfy requirements for the shortest and smoothest path in static or dynamic environments.

In this study, the previous research that's tried to solve the problem of multiobjective optimization were reviewed. For example, Li and Chou [11] employed self-adaptive learning PSOs to attain three objectives: path length, collision risk degree, and smoothness. In order to improve the search capability of a PSO, this mechanism selects the most suitable search strategies dynamically at various stages of the optimization process. An alternative algorithm for multi-objective mobile robot path planning uses an improved genetic algorithm [12] to find three types of objects for planned paths: length, smoothness, and security. By considering a Mars Rover scenario, Guimari in [13] used A* search algorithm to minimise the path difficulty, danger, elevation and length from a starting point to a destination. A novel multi-objective optimization method that uses the WOA is also presented for planning mobile robot paths [14]. The two distance and smooth path criteria of robot path planning are transformed into a minimisation process. In WOA, the solution fitness is determined by the target and obstacle positions in the environment. A multi-objective approach based on the shuffled frog-leaping algorithm (MOSFLA) was proposed in [15], that is, the natural behaviour of frogs. Three distinct path objectives are considered: safety, length and smoothness. The results are compared with the well-known and most commonly used Non-dominated Sorting Genetic Algorithm II. In [16], they were also explored the natural flashing behaviour of fireflies and proposed the multi-objective firefly algorithm (MO-FA). This method addresses three distinct goals: path safety, length and smoothness (related to energy consumption). Furthermore, eight realistic scenarios are used to calculate the path to test the proposed MO-FA. An elitist multi-objective approach based on coefficients of variation was proposed in [17]. The Intelligent water drops (IWD) algorithm was used in the calculation of the Pareto front. Known as CV-based MO-IWD, this method aims to optimize two goals: path length and safety .To optimise the cost performance index with the multi-objective fuzzy optimisation strategy and Voronoi diagram method, A new approach was developed in the [18] to improve the performance index, which can coordinate the weight conflict of the sub-objective functions. A global path planning algorithm for wheeled robots that utilizes multiobjective memetic algorithms (MOMA) that optimizes several objectives simultaneously, including path length, smoothness, and safety was discussed in [19]. Two MOMA are utilized based on conventional multi-objective genetic algorithms combined with elitist, non-dominant sorting and decomposition strategies for optimizing the path length and reducing smoothness simultaneously. To improve the algorithms search capabilities and to ensure the safety of the obtained candidate paths, new path encoding schemes, path refinements, and specific evolutionary operators are designed and introduced in the MOMA. A new method based on Region of Sight is also presented in [20] with the primary goal to easily and rapidly address path planning, the method attempts to meet various robot movement requirements, such as shortest path length, safety and smoothness, while using the least amount of time. An optimization algorithm known as using grey wolf optimization algorithm (GWO) [21] is introduced to solve the problems with multiple objectives. GWO integrates a fixed-sized external archive to save and retrieve the pareto optimal solutions, then define the social hierarchy and simulate the hunting behavior of grey wolves in a multi-objective search space.

Hybridisation is also used to improve the performance of path planning meta-heuristic algorithms. With hybridisation, the benefits of two or more meta-heuristic algorithms are combined to create a better technique. A hybrid multi-objective based on the bare-bones PSO with differential evolution was illustrated in [22]. To achieve different types of optimisation, Geetha *et al.* in [23] proposed a path planning algorithm based on Ant Colony Optimisation and Genetic algorithm. The objectives for planned paths include length, smoothness and security. Ajeil *et al.* [24] optimised paths by conducting a hybridised PSO-modified frequency bat (PSO-MFB) algorithm. A path was optimized using the hybridisation of Grey Wolf Optimiser–PSO algorithm in [25]. Oleiwi *et al.* in [26] proposed new hybrid approach based on enhanced genetic algorithm based on the modified search A* algorithm and fuzzy logic.

However, these studies are mainly focus to solve single /multi objective but they did not take into account the change of location of the target. In reality, the target location commonly changes in time. For example, unmanned aerial vehicles flow target.

This paper presents a novel path planning algorithm to solve multi-objective optimisation path planning problems with moving targets in unknown dynamic environments. The new path planning algorithm achieves safety, shortness, and smoothness. It consists of three modules as follows; firstly, a combination of BA and PSO algorithms is used to select points that are short and smooth enough to satisfy the proposed multi-objective measures. The PSO is used to optimise the BA parameters, which generates the path points. Secondly, to make the infeasible solutions feasible, new LS is used. Thirdly, to achieve the objective, the novel algorithm is applied to detect and avoid the obstacles.

To the best of our knowledge, this is the first use of the hybrid BA and PSO algorithm in multi objective robot path planning with moving target. The remaining of the paper is organized as follows; the problem statement is presented in section 2. Section 3 shows the details of swarm intelligence and the proposed approaches are presented in section 4. In section 5, the work of obstacle detection and obstacles avoiding is presented. The simulation results are discussed in section 6 and the main conclusions are presented in section 7.

2. PROBLEM STATEMENT AND ASSUMPTIONS

Consider a workspace in which a mobile robot is in a start position , and must reach a goal position. This robot referred to as the target for another robot referred to as the second robot. The second robot in a different start position must reach the target robot. Mobile robots' workspaces are assumed to contain a variety of static and dynamic obstacles. The purpose of path planning is to determine the optimal or near-optimal paths (which are the safest, shortest, and smoothest) for the second mobile robot to follow in order to reach the target robot. Several assumptions in this study require clarification to avoid colliding with environmental obstacles.

- The environment represents as room that contains varying shapes of dynamic and static obstacles with different shapes and sizes.
- The two mobile robots are considered as points given their physical bodies, as shown in Figure 1.
- The motion of mobile robots is not affected by kinematic constraints.
- Both robot are free to move in any direction at any time.
- The first robot must be avoiding obstacle but the second robot not only avoid obstacle it must flow the best path according to the criteria (safety, smoothness, shortest).

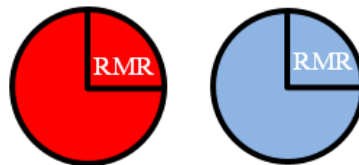


Figure 1. Physical mobile robots

2.1. Performance objectives of mobile robots

The primary goal of path planning is to generate a set of reference points that will carry a robot from a starting point to a destination point. In addition, this path is collision free while trying to meet predefined optimization criteria. Some of these criteria are; maximum safety, lowest energy consumption, shortest path, shortest time, and smoothest path. The optimization criteria can be details as following:

2.1.1. Shortest path

Basically, this is the shortest distance between the starting point and the destination, calculated by summarizing the midpoints ($p(1), p(2), \dots, p(N)$) produced by a path planning algorithm between a Start Point that is given by $p(1)$ and the Goal Point that is denoted by $p(N)$. Figure 2 illustrates this measurement in detail. The (1):

$$f1(x, y) = d(p_i(t), p(N)) \quad (1)$$

$d(.,.)$ corresponds to Euclidean distance between two points. Taking the sum of all distances from the Start Point $p(1)$ to the Goal Point $p(N)$, the shortest path length (SPL) for an exercise is calculated.

$$SPL = \sum_{i=1}^{N-1} d(p_i(t), p_i(t+1)) \quad (2)$$

The index i represent the best solution found by the path planning algorithm.

$$d_t = \sqrt{(x_{p_{i+1}(t+1)} - x_{p_i(t)})^2 + (y_{p_{i+1}(t+1)} - y_{p_i(t)})^2}$$

2.1.2. Path smoothness

Its consider one of the important optimization criteria. The smoothness of the path is normally measured by reducing as much as possible the angle difference between the current goal and the current position produced. Two lines intersect at this angle, which is then used to determine the robot's location at the conclusion of the process.

$$f_2(x, y) = \sum_{t=1}^{N-1} |\theta_{(p_i(t), p_i(t+1))} - \theta_{(p_i(t), p(N))}| \tag{3}$$

where,

$$\theta_{(p_i(t), p_i(t+1))} = \tan^{-1} \frac{y_{p_i(t+1)} - y_{p_i(t)}}{x_{p_i(t+1)} - x_{p_i(t)}} \qquad \theta_{(p_i(t), p(N))} = \tan^{-1} \frac{y_{p(N)} - y_{p_i(t)}}{x_{p(N)} - x_{p_i(t)}}$$

A detailed description of this measurement details in Figure 2 and Figure 3.

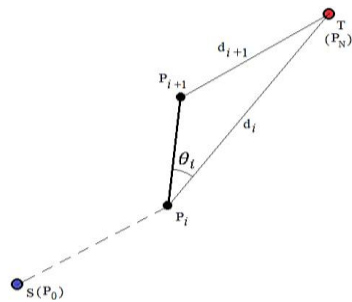


Figure 2. Measurement description

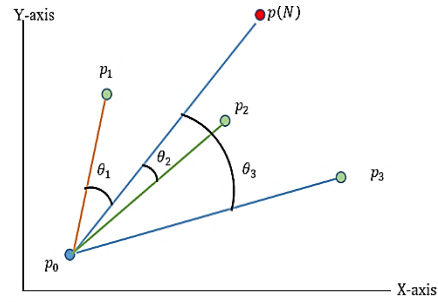


Figure 3 Candidate solutions angles at iteration t

Figure 3 shows that:

θ_1 represents the angles between line segments $(p(0), p(N))$ and $p(0), p(1)$;

θ_2 represents the angles between line segments $(p(0), p(N))$ and $p(0), p(2)$;

θ_3 represents the angles between line segments $(p(0), p(N))$ and $p(0), p(3)$.

Clearly, among the competing points (p_1, p_2, p_3) , p_2 has the minimum angle θ_2 . Overall, multi-objective optimization is a balance between distance and angle of path objectives discussed above.

However, finding the candidate with both advantages is difficult, and thus the two objectives are combined using the weight to calculate the fitness value, as

$$f = (x, y) = w_1 f_1(x, y) + w_2 f_2(x, y) \tag{4}$$

The values of w_1 and w_2 represent the relative importance of the two objectives, respectively. The following requirements must be met by their respective values:

$$w_1 + w_2 = 1 \tag{5}$$

The overall fitness function is defined as shown in (6):

$$fitness = \frac{1}{f(x,y)+e} \tag{6}$$

The e factor prevents division by zero (e.g., $e = 0.001$). In each iteration, an optimal solution among competing options can be selected by determining the balance of the two performance objectives stated in (1) and (3). Figure 4 shows that the best point among four competing points is p_2 for the t and p_3 for the $(t + 1)$ iterations, on the other hand, in the second iteration $(t + 2)$ the distances are shorter for p_1 and p_4 , but the

angles are larger, therefore p2 is chosen because it strikes a balance between the two criteria. This procedure is repeated until the GP is located.

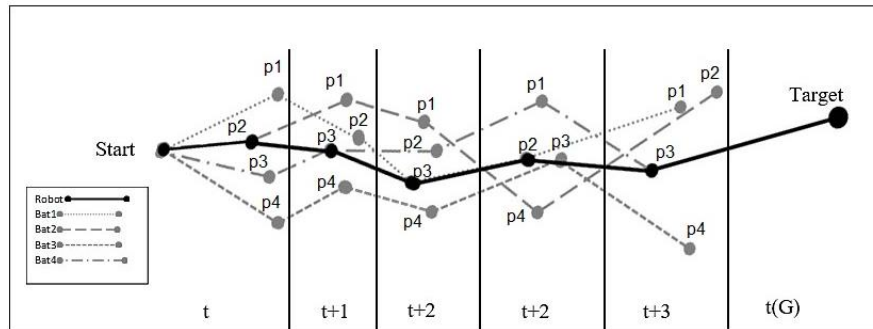


Figure 4. Multi-objective path planning using mid-point selection

In our proposed system, the distance and angle are normalised below to improve the method performance, $\text{new_distance} = \text{distance}/ST$, $\text{new_angle} = \text{angle}/90$, Where ST is the distance from the starting point to the destination. The distance varies in the range $[0, ST]$, and is thus normalised by ST . The angle varies in the range $[0, +90]$, and is thus normalised by 90° . Near the target, the normalised distance is nearly 0 while the normalised angle is in the range $[0, 1]$. In this case, the effect of the angle is greater than that of the distance, preventing the robot from arriving at the target and merely vibrates around it. Therefore, distance normalisation is slightly modified to avoid this phenomenon. In other words, the normalised distance is in the range $[0.3, 1.0]$. Subsequently, although the robot is near the target, the normalised distance is greater than 0 and the above phenomenon can be avoided.

2.1.3. Movement of obstacles

In dynamic environment there are moving obstacle. During each time step, the position of the obstacle changes. According to this study, dynamic obstacles move linearly, that is, along a straight line, with a velocity. v_{obs} and direction (φ_{obs}) defined by (7) and (8).

$$\text{new}x_{obs} = \text{old}x_{obs} + v_{obs} \times \cos \varphi_{obs} \quad (7)$$

$$\text{new}y_{obs} = \text{old}y_{obs} + v_{obs} \times \sin \varphi_{obs} \quad (8)$$

where φ_{obs} is the slope of the linear motion.

3. SWARM INTELLIGENCE MODELS

Swarm intelligence models are referred to as mathematical representations of natural swarm systems. Swarm intelligence models have been proposed in literature and been successfully applied in various real-world scenarios. In the present study, two such algorithms are used to solve the path-planning problem which are PSO and BA algorithms.

3.1. PSO algorithm

Inspired by swarm intelligence, Eberhart and Kennedy [27] developed PSO as an optimisation algorithm in 1995 to mimic the behaviour of social animals. A flock of birds that flies to find food does not need leaders, but rather follows the member closest to the food. Therefore, the flock of birds receives the solution they require through effective communication with members of the population. PSO algorithms also use particles to represent candidate solutions, each with a position x_i and velocity v_i corresponding to the candidate. Position refers to the solution offered by a particle, and velocity refers to how rapidly it moves relative to its current position. Here, both values (position and velocity), are randomized and the PSO constructs the solution in two phases [28]. The velocity of each particle is updated by (9).

$$v_i(t + 1) = v_i(t) + c_1 r_1 (cpbest_i - x_i) + c_2 r_2 (gbest - x_i) \quad (9)$$

where $v_i(t)$ represents the particle velocity, $x_i(t)$ represents the particle position, $pbest$ represents the best position of the individual, $gbest$ represents the best position in the group, $c1$ and $c2$ represent the constant and r represents the random number $[0, 1]$.

The particle position is updated by (10).

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (10)$$

3.2. Bat algorithm (BA)

Yang and Gandomi developed a bio-inspired algorithm in 2010 called BA [29]. The micro bat uses echolocation or bio sonar to locate prey. Echolocation is a major aspect of bat behaviour: bats use sound pulses to detect obstacles as they fly. By using the difference in time between their ears, they detect obstacles. Loudness of the response, and delay time, a bat can determine the speed, size, and shape of prey and obstacles. A bat can also change its sonar in this way by sending high-frequency sound pulses. This allows the bat to gather detailed information about its surroundings in a shorter period of time [30].

3.2.1. Artificial bat movement

Each bat is associated with a velocity $v(t)$ and a location $x_i(t)$ at iteration t , in a D -dimensional search or solution space. Among all the bats, there exists a current best solution. Three rules are used to update the positions and velocities. The bat's position is updated in the following manner:

$$f_i = f_{min} + (f_{max} - f_{min}) * \beta_i \quad (11)$$

$$v_i(t + 1) = v_i(t) + (x_i(t) - x^*)f_i \quad (12)$$

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (13)$$

where β is a uniformly distributed random vector $\in [0, 1]$ and x^* represents the current global best location (solution), which is determined by comparing all solutions among all the number of bats. In each bat, after a solution for LS stage is chosen, a new solution is generated using the random walk principle,

$$x_{new} = x_{old} + \sigma \epsilon A(t) \quad (14)$$

Here ϵ is a random number $\in [-1, 1]$ and represents the intensity and direction of the random walk. $A(t)$, describes the average level of loudness for all bats at step t in an iteration. It is possible to control the step size more effectively by providing a scaling parameter σ .

3.2.2. The loudness and emission of pulses

In order to provide an effective mechanism to control the exploration and exploitation and switch to exploitation stage, when necessary, we have to vary the loudness A_i and pulse rate r_i . After each iteration, the level of loudness and pulse rate needs an update. When a bat has located its prey loudness usually decreases, whereas the rate of pulses is increased in accordance with these (15) and (16).

$$A_i(t + 1) = \alpha A_i(t) \quad (15)$$

$$r_i(t + 1) = r_i(0)[1 - \exp(-\gamma t)] \quad (16)$$

where $0 < \alpha < 1$ and $\gamma > 0$.

4. PROPOSED METHOD

This section describes the the path-planning algorithm as an optimization technique that can help the robot to find its path from the initial position to goal position in static and dynamic environments with moving target. This section discusses the algorithm for planning the path of a mobile robot. It relies on hybrid swarm optimization which consist of PSO and BA algorithms combined with local search, and obstacles detection and avoidance techniques.

4.1. Hybrid PSO-BA algorithm proposed

A hybridised optimisation algorithm is formed by combining the best characteristics of two or more optimisation algorithms to improve the overall performance. Given the limited exploration capability of BA,

convergence of the obtained solutions to the global optimum point is quite impossible [30]. BA contains the bat-specific factors A and r that have an effect on all dimensions of solutions. In this study, a hybridisation of PSO and BA is proposed to optimise the two parameters of the latter and achieve a balance between exploration and exploitation. The proper control of these parameters is based on two variables: alpha (α) and gamma (γ), and thus its optimal value enhances the variation of loudness A_i and pulse rate r_i . Thus, Exploration and exploitation can be balanced with BA. The PSO algorithm adapts the appropriate range (α, γ), Algorithm 1 illustrates the pseudo code for Hybrid PSO–BA while Figure 5 illustrates the overall procedure. The particle size of PSO repressed in two-dimensional vector denoted as $x(\alpha, \gamma)$.

Algorithm 1: A pseudo-code for the hybrid PSO–BA algorithm.

- 1- Initialise PSO and BA parameters:
 PSO_parameters: population size of particles NP, r1, r2, c1, c2
 BA_parameters: population size of bats NB, frequency (f_i), pulse rate(r_i) and loudness (A_i)
- 2- Generate random solutions of PSO $s1 = [\alpha_1, \gamma_1]$, $s1 = [\alpha_2, \gamma_2]$, ..., $s_n = [\alpha_n, \gamma_n]$
- 3- For $i = 1$ to NP
- 4- Use BA algorithm from Equations (11)-(16)
- 5- Pick the bat with the best fitness according to Equation (6)
 $x_{k,j}$ where $j = 1:NB$
- 6- Store index (k) of the best bat
- 7- $Gbest = x_k = [\alpha_k, \gamma_k]$.
- 8- If the stopping criteria is not satisfied, then
- 9- Update position and velocity using Equations (9)-(10) and go to step 3
- 10-Otherwise Calculate the result output
- 11- End if

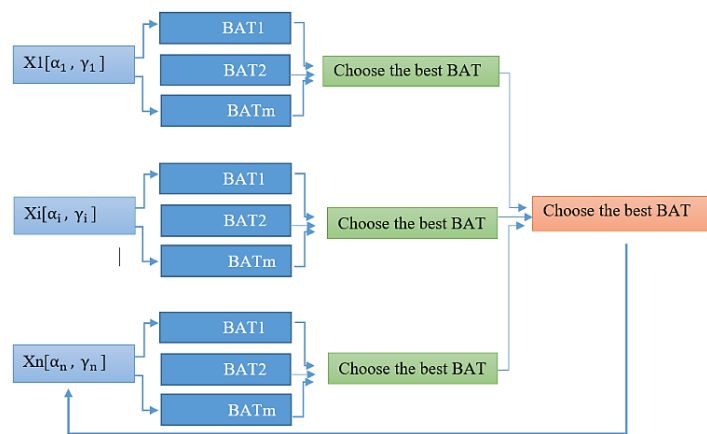


Figure 5. Algorithm proposed for hybrid PSO–BA optimisation

4.2. The proposed LS technique

In every iteration, new positions for the robot are obtained using PSO–BA. However, the position can be inside obstacle (case 1) or the path from the previous to the current position can intersect the obstacle (case 2). Figure 6 shows this scenario, where the local search (LS) algorithm is used to convert the infeasible generated solution in to feasible solution. If the algorithm produces the next point in one of two ways, the solution is considered infeasible:

- In the first case, the points are inside an obstacle
- A segment of a line passes through an obstacle.

4.2.1. Candidate feasibility checking

Two cases consider the candidate solution is infeasible. The first case is when the candidate is inside obstacle and the second is when the line segment between the current and candidate positions intersects the obstacle. However, the two cases have a common feature: grid points are shown near the line segment between current and the candidate positions. Figure 6 shows the two situations of infeasible solution, where p is the current position, and $p1$ and $p2$ are the candidate positions, the neighbouring area is shown by the red outlines, and can thus be considered as one situation that can be explained by:

$$Wpx \leq Ox \leq Wpix$$

$$Wpy \leq Oy \leq Wpiy$$

$$dy \leq s$$

Thus, the above condition is used for checking the candidate feasibility. Where Ox and Oy represent the grid point of the obstacle, Wpx and Wpy are the x, y-coordinates of the Wp , $Wpix$ and $Wpiy$ is for Wpi , s is the scale of grid, dy computed in (17)

$$dy = |Oy - (a \cdot Ox + b)| \quad (17)$$

where a and b are the slope and offset of the line from Wp and Wpi . In other words, the line that passes both Wp and Wpi can be defined as (18).

$$y = a \cdot x + b \quad (18)$$

when O is on the line, then

$$Oy = a \cdot Ox + b \quad (19)$$

However, O may not be on the line but rather on the neighbouring areas. Thus,

$$Oy \approx a \cdot Ox + b$$

The size of the neighbour is the grid scale s .

$$dy = |Oy - (a \cdot Ox + b)| \leq s \quad (20)$$

4.2.2. Modify the infeasible candidate

The LS technique is used to convert these infeasible solutions to feasible ones. This section illustrates the proposed solutions of the two previous types of situations using graphical and mathematical representations. Figure 7 shows that $Wp1$ is inside the obstacle and $Wp2$ is outside the obstacle but the line segment passes between the current position and the destination. First, find the nearest grid point from Wp in the neighbour area of the line segment. In other words, in the Figure 7, $Wq1$ is the nearest point in the neighbouring area of the line segment between Wp and $Wp1$ for the first case, and $Wq2$ for the second case.

$$Wqi = \min_{O \in N(Wp, Wpi)} |Wp - O| \quad (21)$$

$N(Wp, Wpi)$ is the neighbouring area and $|Wp - O|$ is the distance between Wp and O , $i = 1, 2$. Second, calculate the new feasible candidate from the infeasible one.

$$\overline{Wpi} = Wp + (Wpi - Wp) \cdot \frac{|Wp - Wqi| - ds}{|Wp - Wpi|} \quad (22)$$

where ds is the safety distance. In other words, the distance between Wqi and \overline{Wpi} .

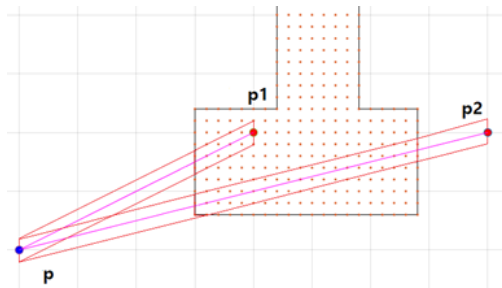


Figure 6. Infeasible candidate solution

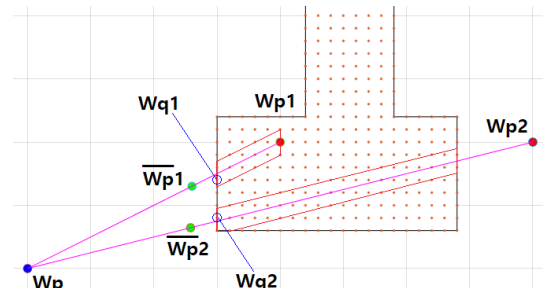


Figure 7. Modifying infeasible positions

5. OBSTACLE DETECTION AND AVOIDANCE (ODA)

The robot follows a path generated by PSO-BA algorithm between the starting point and a goal point until it detects an obstacle. The starting point is denoted by (SP) and the goal point is denoted by (GP). In this case, the robot uses ODA procedure to detect and avoiding the obstacle. The obstacle avoiding (OA) based on wall following procedure.

5.1. Obstacle detection

When planning the path of a mobile robot, the first and most important aspect is to detect obstacles and avoid them. Sensors enable the robot to detect obstacles, and so it is necessary that the robot uses this sense to determine which path to follow. By deploying sensors around a robot, obstacles can be detected. All of them have the same Sensor Range (SR) and Angle. The SR represents the maximum distance that the mobile robot can measure, and the angle of the sensor is calculated by $(360^\circ/\text{no of sensors})$. As obstacles are detected using sensor vectors (VS), which are binary vectors containing information about obstacles, their dimensions depend on the number of sensors (in this case, 12 sensors are used), therefore, the sensor vectors are represented by 12 bits indexed by $i \in \{1, 2, \dots, 12\}$. The value of VS is $[s(1), s(2), \dots, s(12)]$. The value of $s(i)$ is either 1 which indicates that an obstacle is inside the sensing range, or 0 which indicates that the angle range is free as shown in Figure 8.

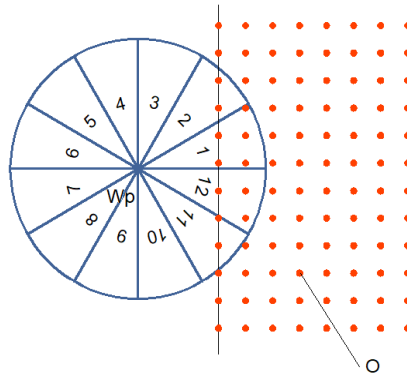


Figure 8. Obstacle detection

The distance between the current position Wp and the obstacle is (23).

$$D = \min_{O \in \text{Obstacle}} |Wp - O| \quad (23)$$

where O is a grid point of the obstacle. If $D \leq sr$, then the obstacle is detected, where sr is the sensing radius. For the grid points inside the SR, calculate the angle of the line segment between Wp and the considered grid point O .

$$\alpha = \text{atan} \left(\frac{Ox - Wpx}{Oy - Wpy} \right) \quad (24)$$

Subsequently, which sensor point O is in the SR can be determined using angle α . Thus, the sensing vector can be calculated. The sensing vector consists of bits, which is the sensor number. That is, each bit corresponds to each sensor. When the i -th sensor detects the grid point, the i -th bit is 1, and otherwise 0.

$$Vs(i) = \begin{cases} 1, & \text{sensor } i \text{ detects the grid point} \\ 0, & \text{else} \end{cases}$$

In the Figure 8, the sensing vector is $Vs = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1]$. In other words, the obstacle is sensed by sensors 1, 2, 11 and 12. Next, the gap vector, which indicates the direction the robot can pass through, is calculated from the sensing vector as $Vg(i) = \text{NOT}(Vs(i) \text{ OR } Vs(i + 1))$.

In the Figure 8, the corresponding gap vector is $Vg = [0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0]$.

5.2. Obstacle avoidance

In order to complete the path to reach the goal avoidance procedure is performed once the robot detects an obstacle. Sensors enable the robot to detect obstacles, and so it is necessary that the robot uses this sense to determine which path to follow. The avoidance procedure is performed using the wall-following algorithm. The mechanism of this algorithm explains as following:

- Move to left or right direction along the ‘wall’ of the obstacle. If the obstacle is detected, the robot moves along the obstacle in the left or right direction. Figure 9 shows the left and right directions for following the wall.
- Detect the wall of the room and change direction

While moving, the robot can detect the boundary of the room using the following conditions:

$$\begin{aligned}
 |Wpx - xmin| &\leq sr \\
 |Wpx - xmax| &\leq sr \\
 |Wpy - ymin| &\leq sr \\
 |Wpy - ymax| &\leq sr
 \end{aligned}$$

Where Wpx and Wpy are the coordinates of the current robot position. Figure 10 shows the room boundary. The range of the room is xmin, xmax in the x-axis and ymin, ymax in the y-axis. Once the room wall is detected, the robot must change direction. This procedure details in Figure 11.

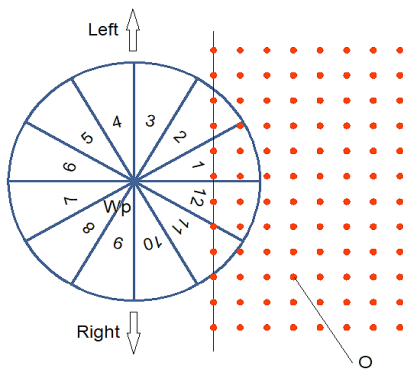


Figure 9. Wall-following algorithm

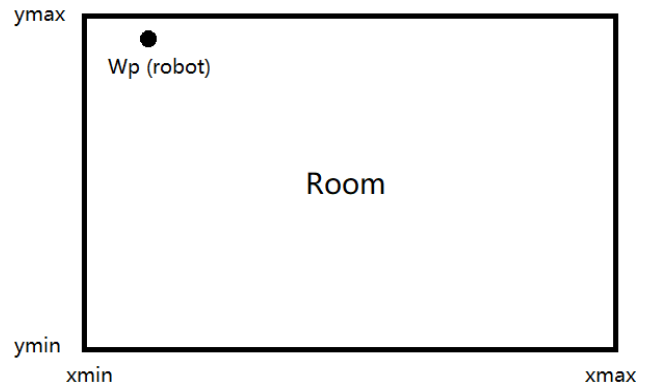


Figure 10. Room boundary

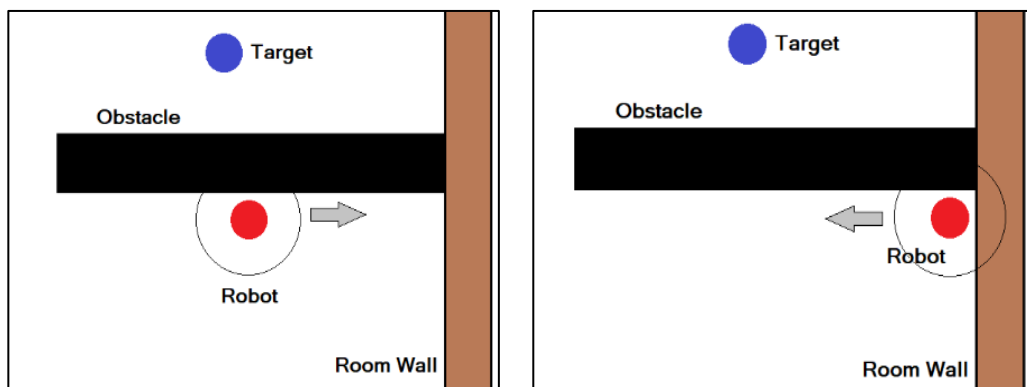


Figure 11. Changing movement direction

- Moving towards the target

When no obstacle is detected, the robot can move towards the target. Figure 12 show moving the robot towards the target. An exception can occur, as shown in Figure 13. In this case, no obstacle is detected in the direction towards the target but the robot cannot actually pass through the path. This case can be avoided by memorizing the robot path during wall following. In the Figure 9, the orange points are those where the robot

passed. Therefore, when passing the points in the direction towards the target, the robot continues to follow the obstacle wall. Otherwise, the robot moves directly towards the target as shown in Figure 14.

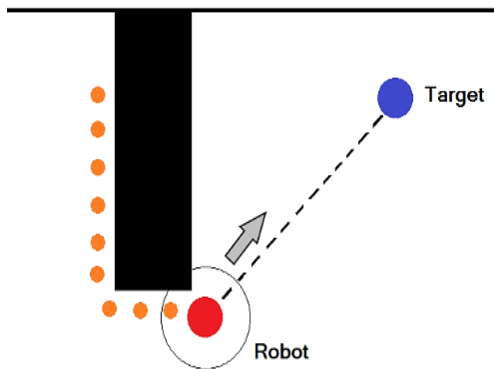


Figure 12. Moving towards the target

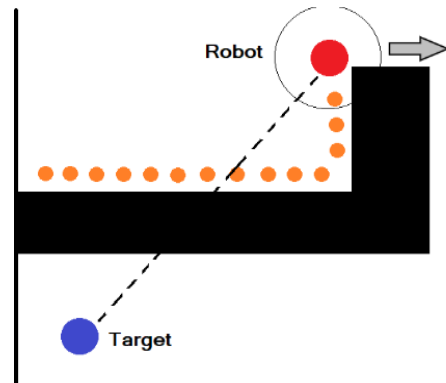


Figure 13. Exception case

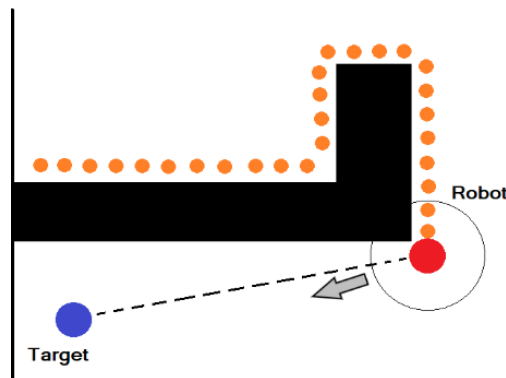


Figure 14. Robot moving directly to the target

6. SIMULATION RESULTS

The algorithm executed with MATLAB R2020a programming language, the following results are obtained as a sample of difference scenario of experiences. MATLAB code is run on a computer with a 2.80 GHz Core i7 processor and 16 GB of RAM. Figure 15(a)-(f) shows the simulation results shown in the appendix. A blue point represents the first mobile robot (Robot1) at the starting point $SPR1 = (120, 120)$. The purple point represents the target point for Robot 1, which is $(20, 60)$. The second robot (Robot2) represented by red point at the location $(160, 5)$. The first robot (Robot1) attempts to reach its target, while the second robot (Robot2) reaches the meeting point with the first robot (Robot1). Several dynamic obstacles with different shapes are moving linearly in this environment, and six static obstacles have different forms and dimensions. The colour of dynamic obstacles is red while the colour of static obstacles is black. As shown in Figures 15(b) to (e) the first robot moving towards its fixed target, while the second robot attempting to move towards the first robot (moving target) while all dynamic obstacles also move in a linear manner. Figure 15 (f) shows the 88th (final) iteration, where the second robot reaches the target without collision with any obstacles in the environment.

7. CONCLUSION

This study proposed a path planning algorithm for mobile robots using a hybrid PSO-BA optimization algorithm. PSO-BA is integrated with the LS algorithm and ODA algorithm. LS is used to covert all infeasible points into feasible points, while the ODA algorithm is used to detect and avoid obstacles. The performance of the algorithm is tested in dynamic environments with moving targets in different scenarios. Using simulation results, the proposed algorithm demonstrates its effectiveness in avoiding both static and dynamic obstacles.

8. APPENDIX

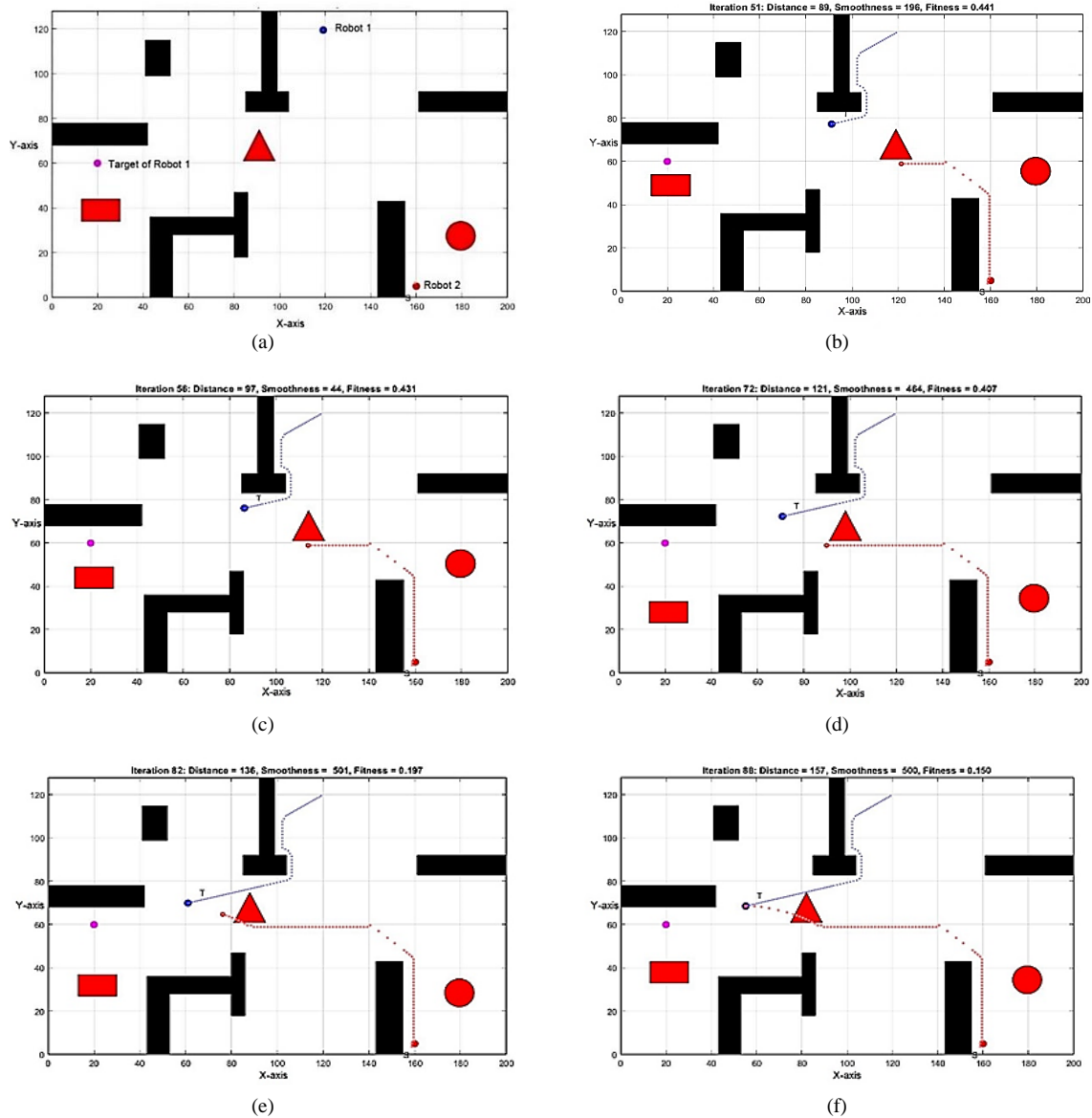


Figure 15. Path planning history: (a) first iteration, (b) Iteration no. (51), (c) Iteration no. (56), (d) Iteration no. (72), (e) Iteration no. (82), (f) Iteration no. (88)




REFERENCES

- [1] W. Jasim and D. Gu, " H_∞ path tracking control for quadrotors based on quaternion representation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8717 LNAI, pp. 72–84, 2014, doi: 10.1007/978-3-319-10401-0_7.
- [2] L. B. Amar and W. M. Jasim, "Hybrid metaheuristic approach for robot path planning in dynamic environment," *Bull. Electr. Eng. Informatics*, vol. 10, no. 4, pp. 2152–2162, 2021, doi: 10.11591/EEI.V10I4.2836.
- [3] H. Cheon and B. K. Kim, "Online Bidirectional Trajectory Planning for Mobile Robots in State-Time Space," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4555–4565, 2019, doi: 10.1109/TIE.2018.2866039.
- [4] S. Hosseinijad and C. Dadkhah, "Mobile robot path planning in dynamic environment based on cuckoo optimization algorithm," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, pp. 1–13, 2019, doi: 10.1177/1729881419839575.
- [5] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Appl. Soft Comput. J.*, vol. 59, pp. 68–76, 2017, doi: 10.1016/j.asoc.2017.05.012.
- [6] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Rob. Auton. Syst.*, vol. 64, pp. 137–141, 2015, doi: 10.1016/j.robot.2014.07.002.
- [7] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A bat algorithm with mutation for UCAV path planning," *Sci. World J.*, vol. 2012, 2012, doi: 10.1100/2012/418946.
- [8] W. Wang, M. Cao, S. Ma, C. Ren, X. Zhu, and H. Lu, "Multi-robot odor source search based on Cuckoo search algorithm in ventilated indoor environment," *Proc. World Congr. Intell. Control Autom.*, vol. 2016-Sept, pp. 1496–1501, 2016.




- doi: 10.1109/WCICA.2016.7578817.
- [9] H. S. Dewang, P. K. Mohanty, and S. Kundu, "A Robust Path Planning for Mobile Robot Using Smart Particle Swarm Optimization," *Procedia Comput. Sci.*, vol. 133, pp. 290–297, 2018, doi: 10.1016/j.procs.2018.07.036.
 - [10] P. K. Das, S. K. Pradhan, S. N. Patro, and B. K. Balabantaray, "Artificial immune system based path planning of mobile robot," *Stud. Comput. Intell.*, vol. 395, pp. 195–207, 2012, doi: 10.1007/978-3-642-25507-6_17.
 - [11] G. Li and W. Chou, "Path planning for mobile robot using self-adaptive learning particle swarm optimization," vol. 61, no. May, pp. 1–18, 2018, doi: 10.1007/s11432-016-9115-2.
 - [12] J. Hu and Q. Zhu, "Multi-objective mobile robot path planning based on improved genetic algorithm," *2010 Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2010*, vol. 2, pp. 752–756, 2010, doi: 10.1109/ICICTA.2010.300.
 - [13] F. G. Guimari, "Multi-Objective Mobile Robot Path Planning Based on," no. Icccke, pp. 7–12, 2016.
 - [14] T. K. Dao, T. S. Pan, and J. S. Pan, "A multi-objective optimal mobile robot path planning based on whale optimization algorithm," *Int. Conf. Signal Process. Proceedings, ICSP*, vol. 0, pp. 337–342, 2016, doi: 10.1109/ICSP.2016.7877851.
 - [15] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "MOSFLA-MRPP: Multi-Objective Shuffled Frog-Leaping Algorithm applied to Mobile Robot Path Planning," *Eng. Appl. Artif. Intell.*, vol. 44, pp. 123–136, 2015, doi: 10.1016/j.engappai.2015.05.011.
 - [16] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach," *Soft Comput.*, vol. 21, no. 4, pp. 949–964, 2017, doi: 10.1007/s00500-015-1825-z.
 - [17] S. Salmanpour, H. Monfared, and H. Omranpour, "Solving robot path planning problem by using a new elitist multi-objective IWD algorithm based on coefficient of variation," *Soft Comput.*, vol. 21, no. 11, pp. 3063–3079, 2017, doi: 10.1007/s00500-015-1991-z.
 - [18] Y. Wang, T. Wei, and X. Qu, "Study of multi-objective fuzzy optimization for path planning," *Chinese J. Aeronaut.*, vol. 25, no. 1, pp. 51–56, 2012, doi: 10.1016/S1000-9361(11)60361-0.
 - [19] Z. Zhu, J. Xiao, J. Q. Li, F. Wang, and Q. Zhang, "Global path planning of wheeled robots using multi-objective memetic algorithms," *Integr. Comput. Aided Eng.*, vol. 22, no. 4, pp. 387–404, 2015, doi: 10.3233/ICA-150498.
 - [20] H. Hliwa and B. Atieh, "Multi Objective Path Planning in Static Environment using Region of Sight," *Proc. 2nd 2020 Int. Youth Conf. Radio Electron. Electr. Power Eng. REEPE 2020*, pp. 1–5, 2020, doi: 10.1109/REEPE49198.2020.9059199.
 - [21] S. Mirjalili, S. Saremi, and S. Mohammad, "Multi-objective grey wolf optimizer : A novel algorithm for multi-criterion optimization," vol. 47, pp. 2015–2017, 2016.
 - [22] J. H. Zhang, Y. Zhang, and Y. Zhou, "Path planning of mobile robot based on hybrid multi-objective bare bones particle swarm optimization with differential evolution," *IEEE Access*, vol. 6, pp. 44542–44555, 2018, doi: 10.1109/ACCESS.2018.2864188.
 - [23] S. Geetha, G. M. Chitra, and V. Jayalakshmi, "Multi objective mobile robot path planning based on hybrid algorithm," *ICECT 2011 - 2011 3rd Int. Conf. Electron. Comput. Technol.*, vol. 6, pp. 251–255, 2011, doi: 10.1109/ICECTECH.2011.5942092.
 - [24] F. H. Ajelil, I. K. Ibraheem, M. A. Sahib, and A. J. Humaidi, "Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm," *Appl. Soft Comput. J.*, vol. 89, pp. 1–27, 2020, doi: 10.1016/j.asoc.2020.106076.
 - [25] F. Gul, W. Rahiman, S. S. N. Alhady, A. Ali, I. Mir, and A. Jalil, "Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO-GWO optimization algorithm with evolutionary programming," *J. Ambient Intell. Humaniz. Comput.*, no. 0123456789, 2020, doi: 10.1007/s12652-020-02514-w.
 - [26] B. K. Oleiwi, H. Roth, and B. I. Kazem, "Modified Genetic Algorithm based on A* Algorithm of Multi Objective Optimization for Path Planning," *J. Autom. Control Eng.*, vol. 2, no. 4, pp. 357–362, 2014, doi: 10.12720/joace.2.4.357-362.
 - [27] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," *Proc. Int. Symp. Micro Mach. Hum. Sci.*, pp. 39–43, 1995, doi: 10.1109/mhs.1995.494215.
 - [28] A. Koubaa, H. Bennaceur, I. Chaari, S. Trigui, and A. Ammar, *Robot Path Planning and Cooperation*, vol. XXII, no. 2, 2018.
 - [29] X. S. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Eng. Comput. (Swansea, Wales)*, vol. 29, no. 5, pp. 464–483, 2012, doi: 10.1108/02644401211235834.
 - [30] S. Yilmaz, E. U. Kucuksille, and Y. Cengiz, "Modified bat algorithm," *Elektron. ir Elektrotehnika*, vol. 20, no. 2, pp. 71–78, 2014, doi: 10.5755/01.eee.20.2.4762.

BIOGRAPHIES OF AUTHORS



Baraa M. Abed    **Abed** is a Ph.D. Student in the field of Robotics and Machine learning. He has received B.Sc. and M.Sc. degrees in Computer Science from University of Anbar. His research interests are autonomous systems, robotics, machine learning, classification and optimization technique. He has published research papers at national and international journals, conference proceedings. He can be contacted at email: burasoft@gmail.com.



Wesam M. Jasim    **Jasim** received the B.Sc. and M.Sc. degrees in control automation engineering from University of Technology, Baghdad, Iraq, and Ph.D. degree in computing and Electronics from University of Essex, Essex, UK. Currently, he is an Assistant Professor with the College of Computer Science and Information Technology, University of Anbar. His current research interests include robotics, multiagent systems, cooperative control, Robust control, linear and nonlinear control, Deep learning. He has published research papers at national and international journals, conference proceedings as well as chapters of book. He can be contacted at email: co.wesam.jasim@uoanbar.edu.iq