Received: December 6, 2022. Revised: January 9, 2023. 215

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multi-Objective Robot Path Planning Using an Improved Hunter Prey Optimization Algorithm

Jaafar Ahmed Abdulsaheb[1,2]* Dheyaa Jasim Kadhim[1]

[1]*Department of Electrical Engineering, College of Engineering, University of Baghdad, Iraq, Iraq*
[2]*Department of Electronics and Communication, College of Engineering, Uruk University, Iraq*
* Corresponding author's Email: jaafer@uruk.edu.iq

Abstract: This work considers the best path planning algorithm for a mobile robot that travels independently in an unknowable environment. To get around the limitations of unstable searches in the conventional Hunter-Prey Optimization Algorithm (HPO), the improved HPO optimization algorithm is used and introduces a new control parameter referred to as randomization adjustment in order to avoid stagnation and early convergence. The absence of the transfer parameter from exploration and exploitation is a significant flaw in the HPO algorithm, which results in an unstable search and additional time waste. Another new parameter called the changing parameter (CP) is used to address this flaw. It is used in an environment with erratic static and dynamic obstacles and a static and dynamic target. Finding a collision-free path that is also the objectively shortest path and the smoothest path can solve the path-planning problem. The proposed algorithm attempts to mimic the real world by taking into account the actual size of the mobile robot, a kinematic model, and the robot's specifications. The proposed algorithm is evaluated by comparing it on 30 dimensions using 13 benchmark test functions. The performance of the proposed algorithm is evaluated against the results of five swarm optimization algorithms. According to the results of the standard deviation tests, the proposed algorithm performs the best in 92% of the 13 test functions. Furthermore, the average outcomes for three complex maps (10×10) m in size demonstrate the potency of this approach for robot paths from the starting point to the target. The average distance over ten runs for maps 1, 2, and 3 is 12.6436 meters, 12.4961 meters, and 17.6547 meters, respectively. It demonstrated how quickly and easily it could avoid both stationary and moving obstacles.

Keywords: Robot path planning, Multi-objectives optimization, Hunter prey optimization algorithm, Shortest path, Smoothness, Obstacle detection and avoidance.

1. Introduction

Numerous industries, including searches and rescue, the armed services, agricultural production, medical services, and entertainment., use autonomous mobile robot (AMR) navigation [1]. Concerning robot navigation (RN), there are three main problems that need to be solved: safety, accuracy, and speed. The safety and accuracy issues are finding a collision-free path and following the precise addressed path. The ability of the algorithm to stop and turn robots repeatedly is referred to as efficiency. This is a waste of time and effort. Several categories of RN problems include localization, path planning, cognitive mapping, and motion control. Path planning could be argued to be the most important issue. The aim of path planning is to find the best, most direct, and collision-free route from a starting point to an objective in a given environment. A robot can usually get to its goal in more than one way, but the best way is chosen based on a set of rules [2].

Robot path planning (RPP) was first studied in the 1960s, and several strategies, including the cell decomposition approach [3], roadmap [4], and potential fields algorithm [5], have since been put forth. The main drawbacks of the aforementioned methods are their inaccuracy and inefficiency (large processing costs) and a significant risk of getting stuck in relative minima. To get around these algorithms' drawbacks, several heuristic techniques

can be applied [6]. These include genetics, neural networks, and nature-inspired algorithms. The following describes some of the associated works.

To prevent becoming stuck in local minimums or experiencing slower convergence while path planning, it was suggested in [7] to enhance the conventional ACO. The challenge of mobile RPP is addressed in [8] using a novel approach based on adaptive particle swarm optimization (APSO). Real-time problems are frequently solved by the APSO method, which is more intelligent than the conventional PSO algorithm. The robot will hopefully come up with a new way to get around obstacles and move faster. In the work [9], a team of mobile robots introduces a novel cuckoo search-based odor source localization method. It makes use of a robot that determines the location of the maximum gas concentration, from which it directs further robots to search for odor sources upwind. This technique can be used by the robots to escape both eddy locations and localized high-concentration areas. In the study [10], a brand-new algorithm was created using the bacterial foraging optimization (BFO) method. It devises a path to the destination and navigates past impediments using particles that are scattered around the robot in a circle. The work [11] employs the artificial immune algorithm (AIA), which is based on the idea of immunity, to choose a path for mobile robots that avoids obstacles. The outcomes of the simulations demonstrate that the mobile robot can use AIA to avoid hazards, get out of binds, and accomplish its objective. In this study, an obstacle-avoiding path for mobile robots is planned using the artificial immune algorithm (AIA), which was developed from the immune principle. The outcomes of the simulations demonstrate that the mobile robot can use AIA to avoid hazards, get out of binds, and accomplish its objective. It has been suggested to use a unique multi-objective approach based on the Whale optimization algorithm (WOA) to plan the best possible paths for mobile robots [12]. WOA transforms the smoothness and distance of the robot's path planning problems into minimization problems. The robot selects the best whale in each iteration and advances in line toward it. Path optimization issues have seen extensive use of GA [13]. The newly suggested crossover operator prevents early convergence and enables pathways with higher fitness values than their parents. I Robot Create (a mobile robot) features a fuzzy logic controller [14] that interfaces with the arduino uno. The robot's left and right wheels travel at different speeds, which are controlled by fuzzy rules. The hybrid multi-objective bare-bones particle swarm optimization with differential evolution [15] method is used to help mobile robots plan better routes. To choose a particle's individual optimum position, a novel Pareto dominance with collision limitations has been created. The efficiency of this algorithm is supported by simulation data. RPP makes use of and recommends the chicken swarm optimization algorithm (ICSO) (improved) [16]. The numbers demonstrate that the ICSO The approach is more accurate, stable, and has a more powerful search capability in RPP for unconstrained optimization. [17] proposes a path planning algorithm with self-adaptive population size based on the firefly algorithm. Population size distinguishes between viable and infeasible solutions. The suggested method is better than the fixed population size firefly algorithm in terms of how stable it is, how quickly it converges, and how long it takes to calculate. The Morphin algorithm [18] was developed to swiftly dodge moving obstacles. Simulation results show that the proposed method performs well for planning an initial, static optimal path. [19] The authors provide a dynamic window strategy for combining the maximum and minimum ant systems and creating an adaptive distance induction factor based on the improved ant colony method. Simulations show that the technique does what it's supposed to do, which is improve the performance of global path optimization while avoiding local dynamic barriers.

The mobile robot was treated as a simple particle in the studies mentioned above, which is one of their flaws. Some of these algorithms were designed to find the shortest path while dodging static obstacles, but other research concentrated on dodging dynamic obstacles while achieving the shortest distance without taking the path's smoothness into account. The grid-based methods used in some of the aforementioned researches are also simple to use, but they have a number of drawbacks, such as an imprecise representation of the obstacle that reserves the entire cell even if the obstacle only occupies a small portion of the cell. Space is wasted as a result, and environments that are dynamic have less flexibility. Additionally, the robot was depicted as a point, and the actual dimensions of the mobile robot were not taken into account.

There are many modern algorithms that have not been applied to path planning yet but may be in the future. The following describes four of these new algorithms: [20] The guided pelican algorithm (GPA) has the improvements needed for a shortcoming in another algorithm, namely the pelican optimization algorithm (POA). GPA mimics the behavior of pelican birds during hunting. Simulation is implemented to observe GPA's performance in

optimizing both theoretical and real-world problems. [21] The stochastic Komodo algorithm (SKA) is an improved version of the Komodo mlipir algorithm (KMA), which is inspired by the behavior of the Komodo dragon during foraging and mating. The improvement is conducted by simplifying the basic form of KMA. It eliminates the sorting mechanism at the beginning of the iteration. Work [22] proposes a new metaheuristic algorithm: a fixed-step average and subtraction-based optimizer (FS-ASBO). This algorithm is then implemented into a simulation to evaluate its performance. The result shows that this proposed algorithm is competitive in solving theoretical problems and superior in solving real-world problems. Paper [23] aims to introduce a new optimization algorithm called the Puzzle Optimization Algorithm (POA) to solve various optimization problems. The main advantage and feature of the proposed POA is that it has no control parameters and therefore does not require parameter setting.

The principal contributions made by this research project are listed below:

(1) The main problem with the traditional HPO algorithm is that it doesn't include the parameter of transfer from exploitation to exploration. As a result, searches become unstable (the local optimum stagnates), wasting more time. In order to get around this problem, the IHPO algorithm was proposed.
(2) This algorithm is used to generate and choose various multi-objective combinations (shortest path and smoothness), as proposed in this paper. In order to compare the new algorithm to the previous one, thirteen benchmark test functions were also used.
(3) The proposed IHPO algorithm is combined with a local search method that turns impractical solutions into ones that can be used in an uncharted environment with random erratic static obstacles, a static and moving target, and random dynamic obstacles.
(4) Also, the kinematic model with robot specifications and the actual size of the mobile robot are taken into account (assuming turtlebot3 burger as a given).

The structure of this paper is as follows: Problem formulation is represented in section 2. Section 3 discusses the HPO optimization. Section 4 describes our suggested improved HPO algorithm; Section 5 illustrates robot path planning using IHPO; and section 6 displays the simulation result and discussion. Finally, section 7 of this work represents its conclusion.

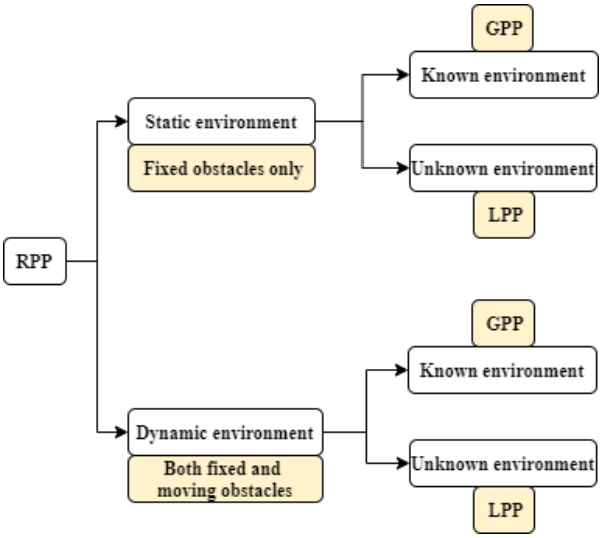

Figure. 1 Classification of RPP

2. Problem formulation

One of the crucial elements in the study of robot navigation is Robot path planning (RPP). As shown in Fig. 1, RPP can be classified into two types based on the environment in which the robot is located: static (environment with fixed obstacles) and dynamic (environment with moving obstacles). There are additional subgroups that can be created within each of these two categories: both local path planning (LPP) and global path planning (GPP), in which all fixed and moving obstacles are known in advance, can be prepared before the robot moves (offline) (LPP). At this location (GPP), it is impossible to get an advance understanding of the environment. In order to gather information about the environment the mobile robot is moving through, there are sensors (online) [24].

In this work, local path planning is used in a setting with erratic static and moving obstacles and an unknowable fixed and moving target. The actual size of the mobile robot as well as the kinematic model with guidelines for robots in 2-D space are also taken into consideration. The MTALAB workspace imports the robot map, and an occupancy binary map is produced. Every pixel on the occupancy map, which is a 2D matrix, either has a binary 0 (empty) or a binary 1 (filled) (occupied by a static or moving obstacle).

3. Standard hunter prey optimization algorithm (HPO)

In 2022, Naruei et al.[25] proposed HPO optimization, a new intelligent optimization algorithm. It simulates the animal hunting process and has the advantages of fast convergence and strong optimization ability. In the standard HPO algorithm, the population position in the solution space is set at random. The formula for setting the population position is as follows:

$$x_i = rand(1, d) \times (u - l) + l \quad (1)$$

x_i denotes the position of the *i*th hunter or prey, *i = 1, 2,... N*, *N* denotes the population size, *l* and *u* are the search space's lower and upper bounds respectively, and *rand*(1,*d*) is the random number of [0, 1], *d = 1, 2,..., M*, and *M* represents the size of the search space. The formula for updating the hunter's location is as follows:

$$x_{j,i}(t+1) = x_{j,i}(t) + 0.5\left[\left(2 \times C \times Z \times P_{pos(i)} - x_{j,i}(t) + 2(1-C)Z\mu_{(i)} - x_{j,i}(t)\right)\right] \quad (2)$$

where x(*t*) and x(*t + 1*) represent the location of hunters now and in the future, respectively; P_{pos} represents the location of the prey. $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the average of all locations; Eq. (4) calculates the adaptive parameter Z.

$$P = r_1 < C;\ IDX = (P == 0) \quad (3)$$

$$Z = r_2 \otimes IDX + r_3 \otimes (\sim IDX) \quad (4)$$

Where *r1* and *r3* are [0, 1] random vectors; *P* is a 0 or 1 random vector; and *r2* is a random number within [0, 1]. *IDX* is the index value of the vector *r1* that meets the conditions (*P* == 0); and *C* is the factor balancing exploitation and exploration, whose value decreases from 1 to 0.02 during the iterative process. The following is an illustration of the calculation:

$$C = 1 - it\left(\frac{0.98}{It_{max}}\right) \quad (5)$$

where it and Itmax represent the number of iterations at present and at the most, respectively. Eq. (6) shows how to figure out the euclidean distance from the average position of each person searched:

$$D_{euc(i)} = \left(\sum_{j=1}^{d}\left(x_{i,j} - \mu_{i,j}\right)^2\right)^{\frac{1}{2}} \quad (6)$$

The search agents that are the furthest away from the average position μ are considered prey Ppos:

$$P_{pos} = x_i \mid i\ is\ index\ of\ Max(\ end\ )\ sort(D_{euc}) \quad (7)$$

The algorithm's convergence is poor if each iteration takes the greatest distance between the search agent and the average μ position into account. When the prey is captured in the actual hunting scene, the hunter will move to the new prey location the next time. To simulate this scenario, the decreasing mechanism is used, as shown in Eq. (8).

$$kbest = round(C \times n) \quad (8)$$

Where n represents how many search agents there are, $kbest = N$ at the start of the algorithm. During each iteration of the algorithm, the hunter chooses the search agent that is farthest from where the prey is usually found and attacks it while the $kbest$ gradually decreases. The $kbest$ is equal to the first search agent at the end of the algorithm (the shortest distance from the average position). So, Eq. (7) can be replaced by Eq. (9) to figure out where the prey is:

$$x_{i,j}(t+1) = T_{pos(j)} + C \times Z\cos(2\pi r_4) \times \left(T_{pos(j)} - x_{i,j}(t)\right) \quad (9)$$

Where x(*t*) and x(*t+1*) represent the prey's current and next iteration positions, respectively; The next prey position is determined by the function *cos* and its input parameters at various radii and angles from the global optimal position, where *Tpos* is the global optimal position, *r4* is a random number within [1, 1], and. When Eqs. (2) and (9) are put together, you can choose the following updated version of the hunter or prey position.

$$x_{i,j}(t+1) = \begin{cases} x_{i,j}(t) + 0.5\left[\begin{matrix}\left(2 \times C \times Z \times P_{pos(j)} - x_{i,j}(t)\right) \\ + \left(2(1-C)Z\mu_{(j)} - x_{i,j}(t)\right)\end{matrix}\right], (10a) \\ T_{pos(j)} + C \times Z \times cos(2\pi r_4) \times \left(T_{pos(j)} - x_{i,j}(t)\right), (10b) \end{cases}$$

r_5 is a random number between 0 and 1, and β = 0.1 is the adjusting parameter If $r_5 < \beta$ the search agent is considered a hunter, the location update formula is (10a); if $r_5 \geq \beta$ the search agent is

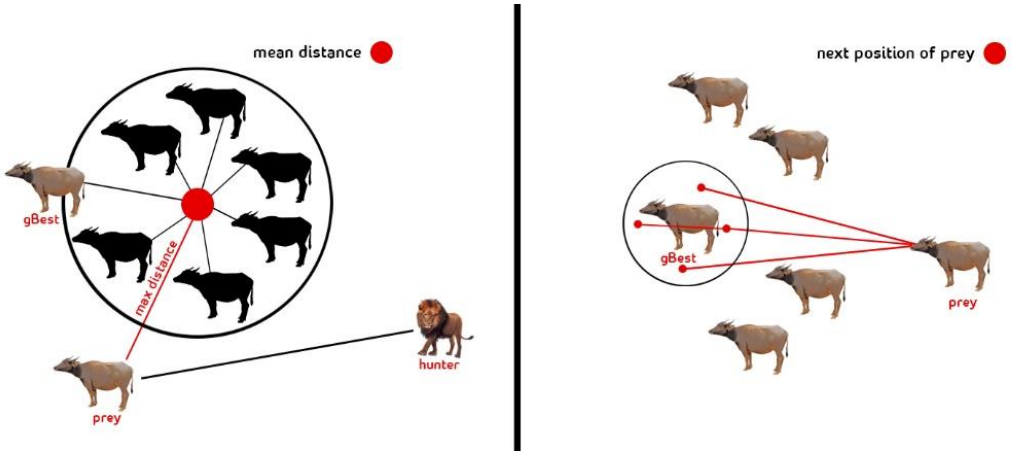

Figure. 2 :(a) Hunter behaviour, and (b) prey behaviours

considered prey, the location update formula is (10b). As shown in Fig. 2.

4. Improved Hunter–prey Optimization (IHPO) Algorithm

1) Any metaheuristic algorithm's main control parameter is the randomization factor. In contrast, we introduced a control parameter called β to adjust the randomization in order to prevent stagnation and early convergence. The HBO algorithm uses rand [0,1] as the only random factor in Hunter–prey 's Eq (10). The newly introduced parameter aids in changing the direction of the search and is defined as follows:

$$\beta = 2 \times rand - 1 \tag{11}$$

2) The absence of the transfer parameter from exploration and exploitation is a significant flaw in the HPO algorithm, which results in an unstable search and additional time waste. A new parameter called the changing parameter (CP) is used to address this flaw. CP, periodic parameter that is used to increase the chance of exploitation and speed up the process of exploration.

$$CP = sin\left(C - \frac{t}{T}\right) \tag{12}$$

$$\text{Let, } \alpha = \beta \times CP \tag{13}$$

$$x_{i,j}(t+1) = \begin{cases} x_{i,j}(t) + 0.5\left[\begin{matrix}\left(\alpha \times 2 \times C \times Z \times P_{pos(j)} - x_{i,j}(t)\right) \\ + \left(2(1-C)Z\mu_{(j)} - x_{i,j}(t)\right)\end{matrix}\right], (14a) \\ T_{pos(j)} + \alpha \times C \times Z \times cos(2\pi r_4) \times \left(T_{pos(j)} - x_{i,j}(t)\right), (14b) \end{cases}$$

5. Robot path planning using IHPO

The path taken by the robot should be optimized to meet certain standards so that the path planning problem can be categorized as an optimization problem. The first objective is to find the shortest path without violating the problem's constraint, which is to avoid all obstacles in the way. The formulation of the optimization problem is a discrete optimization problem in which the objective function $f_1(x,y)$ seeks to minimize the total path traveled by the mobile robot. It is given by:

$$f_1(x,y) = \sum_{j=1}^{n-1} \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{15}$$

As shown in Fig. 3, where n is the number of via points where a change in trajectory happened, which is a project parameter in this case, and:

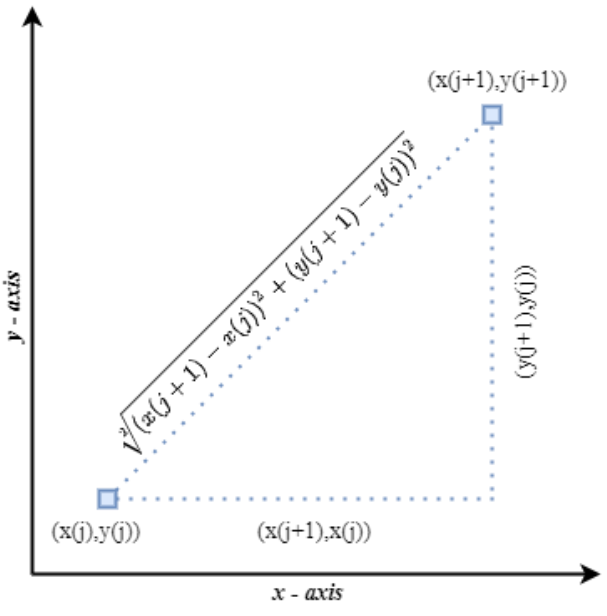

Figure. 3 The Euclidean distance between two points

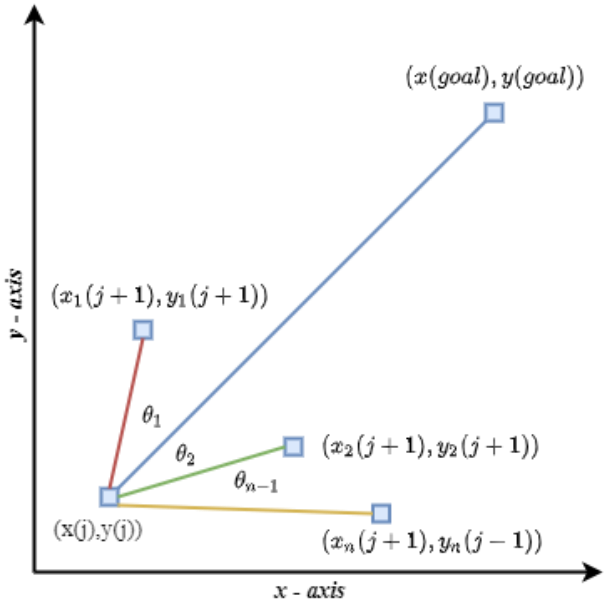

Figure. 4 Path smoothness

$$\Delta x = x(j+1) - x(j) \quad (16)$$

$$\Delta y = y(j+1) - y(j) \quad (17)$$

In addition to the first goal, the algorithm should also meet the second goal, which is to minimize the angles made by the goal, the current position, and the next suggested position's straight lines, as shown in Fig. 4 and the following equation:

$$f_2(x,y) = \sum_{j=1}^{n} \Delta\theta_j \quad (18)$$

$$\Delta\theta_j = tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) - tan^{-1}\left(\frac{y(goal)-y(j)}{x(goal)-x(j)}\right) \quad (19)$$

Where, $j = 1,2,\ldots n-1$

When there are multiple objective functions in a problem that needs to be optimized and the aim is to find one or more optimal solutions, multiple objective optimization (MOO) is used. A popular technique for handling multiple objectives in optimization is the weighted sum method. Fig. 5 shows how the different objective functions are combined into a single objective function that is easier to understand using the weighted sum [24].

$$f(x,y) = \sum_{m=1}^{M} W_m f_m(x,y) = W_1 f_1(x,y) + W_2 f_2(x,y) \quad (20)$$

The weighting coefficient, W = (W_1, W_2,..., W_m), must be defined because it determines the strong solution. These weights have undoubtedly been satisfying and beneficial [26]. $\sum_{m=1}^{M} W_m = 1$, $W_m \in [0,1]$.

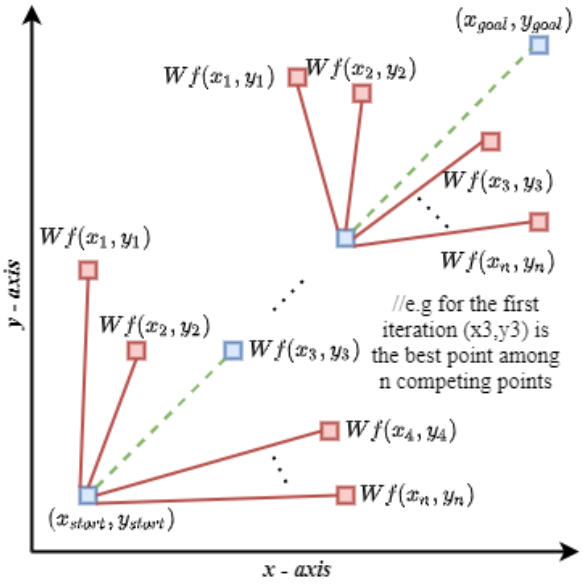

Figure. 5 Weighted sum selections for MOO

The velocity (v_{obs}) and direction (θ_{obs}) of the dynamic obstacles in the case of the dynamic obstacle that shifts from one location to another at each time step, the following equations assume that it is random.

$$\chi_{obs} = \chi_{obs} + v_{obs} \times cos\,\theta_{obs} \quad (21)$$

$$y_{obs} = y_{obs} + v_{obs} \times sin\,\theta_{obs} \quad (22)$$

Where,

$$\theta_{obs} = 360 \times rand(0,1) \quad (23)$$

$$v_{obs} = rand(0,1) \quad (24)$$

Using the same principle from [27] to detect and avoid the obstacles with a local search strategy, the proposed algorithm for path planning can be clarified in the following flowchart, Fig. 6.

6. Results and discussions

6.1 IHPO algorithm performance on benchmark test functions

This section evaluates the IHPO algorithm using 13 criterion functions and 30 dimensions. These are common functions that have been used by many researchers [24]. IHPO compares the results of these tests to those of the HPO algorithms. These common functions are shown in Table 1, where (Range) indicates the limit of the function's search space and (f_{min}) is the ideal value. The last six functions are multimodal, while the first seven are unimodal.

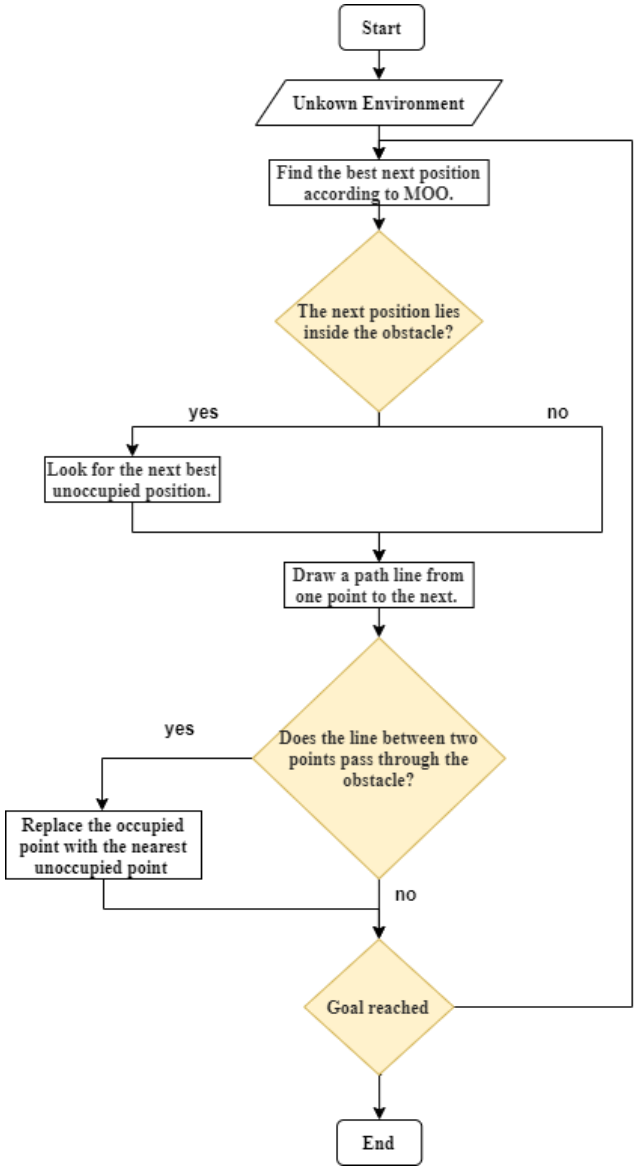

Figure. 6 Flowchart for the proposed algorithm

The fact that the unimodal functions (f1- f7) have a global optimum but no local optimum makes them well suited for use in determining how algorithms should be exploited. Multi-modal functions (f8–13) have a lot of local optimalities. Because of this, they can be used to look at the exploration and avoid the local optima of algorithms.

Optimization has become a popular research subject in recent years, as well as a cost-effective technique to find an ideal solution to complex issues. Five swarm optimization algorithms are compared with the IHPO algorithm: the first is particle swarm optimization (PSO) [28]; the second is another well-known algorithm called the Salp Swarm Algorithm (SSA) [29]; and the fitness dependent optimizer (FDO) [30]. They are also compared with the conventional COOT [31] optimization algorithm and the conventional HPO algorithm to validate their results. The maximum number of iterations is 500, and there are 30 search agents.

The first one is the PSO algorithm. This metaheuristic algorithm borrows social behavior from natural groups of creatures, such as fish schools and bird flocks. It was developed in 1995 by Eberhart and Kennedy and is an optimization tool with a rapidly growing user base for resolving various engineering and scientific issues. The PSO imitates social animal behavior, but it doesn't need a group leader to get the job done. The flock of birds does not need a leader when searching for food; instead, they follow the member who is closest to the food. In this manner, the flock of birds successfully communicates with the other members of the population to arrive at the required solution. The PSO algorithm is made up of a collection of particles, each of which represents a potential resolution.

The second is the SSA algorithm. Salps have a transparent, cylindrical body and are members of the Salpidae family. Their tissues resemble jellyfish tissues in many ways. They also move very similarly to jellyfish, which propel themselves forward by pumping water through their bodies. Because of how challenging it is to access their environments and maintain them in laboratory settings, biological research on this creature is still in its early stages. The swarming behavior of salps is among their most fascinating behaviors. A salp chain is a type of swarm that frequently forms in deep oceans. Although the primary motivation for this behavior is still unclear, some researchers think that it is carried out to improve locomotion through quick, coordinated movements and foraging.

The FDO algorithm is the third. This algorithm mimics the reproductive behavior of a swarm of bees. The core of this algorithm was inspired by how scout bees select a new suitable hive from a large pool of potential hives. In this algorithm, every scout bee that looks for new hives represents a potential solution; in addition, picking the best hive out of several good hives is thought to be convergent to optimality. The search space's artificial scout population is initially initialized randomly by the algorithm; each position of the scout bees represents a recently found hive (solution). Scout bees randomly search more locations in an effort to find better hives; each time a better hive is found, the previous one is disregarded. Similarly, each time the algorithm finds a new, better solution, the previous one will be disregarded. Additionally, the artificial scout bee will continue in its previous direction in the hopes that it will lead it to a better solution (a hive) if the current move fails to do so. The current solution, which is the best one so far, will be continued, though, if the previous

Table 1. Functions for benchmark tests

Function	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	[-100, 100]	0
$f_4(x) = max\{\lvert x_i \rvert, 1 \leqslant i \leqslant n\}$	[-100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_l - 1)^2]$	[-30, 30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	[-100, 100]	0
$f_7(x) = max\{\lvert x_i \rvert, 1 \leqslant i \leqslant n\}$	[-1.28, 1.128]	0
$F_8(x) = \sum_{i=1}^{n} - x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500, 500]	-418.9829×5
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12, 5.12]	0
$F_{10}(x) = -20\,\mathrm{cxp}\left(-0.2\sqrt{\frac{1}{n}\sum_{i-1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i-1}^{n} \cos(2\pi x_i) \right) + 20 + c$	[-32, 32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i-1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i-1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+ \sum_{i-1}^{n} u(x_i, 10,100,4) + \sum_{i=1}^{n} u(x_i, 10,100,4) y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	[-50, 50]	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]\right.$ $\left. + (x_{11} - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	[-50, 50]	0

direction does not result in a better solution. Scout bees randomly look for hives in the wild. This algorithm uses artificial scouts to perform an initial random search of the environment using a random walk and a fitness weight mechanism. The artificial scout bee hopes to discover a better solution each time it moves by accelerating its current position.

The fourth, the Coot algorithm, a swarm-based algorithm inspired by collective movements (irregular and regular movements on the water's surface), was developed by Naruei and Keynia in 2021. A few coots in front of the group act as the group's leader and direct everyone else to the desired location (food). Coots move on the water's surface in

four distinct ways: random movement, chain movement, adjusting position based on group leaders, and leading the group to the optimal location, which is led by their leaders. The last one is a standard HPO algorithm.

These algorithms were applied to 13 test functions with 30 dimensions. Six of the seven unimodal test functions yield the best results when the proposed IHPO algorithm is used, as shown in Table 2. (F1-F7). This shows that the search space has been effectively utilized by the IHPO algorithm. Utilizing multimodal functions, algorithm exploration is measured. In these functions, there are a number of local optimum conditions that the algorithm ought to avoid. The statistical output of the algorithms for these functions is shown in Table 2 (F8-F13). In every multi-modal test function, the new IHPO algorithm performs better. So, the simulation results demonstrated that the proposed algorithm offers significant advantages over the options to which it was compared. According to the results of the standard deviation tests, the proposed algorithm performs the best in 92% of the 13 test functions.

6.2 RPP in a challenging static setting

This section illustrates the effectiveness of the path planning algorithm that has been developed for mobile robots in environments with static obstacles. Six erratic static obstacles of various sizes make up the static environment (Map 1). (0, 0) served as the starting position, and the goal position (10, 10). The following configurations were used with the proposed IHPO algorithm in static environments: The accepted error must be less than 0.2 m, where error is defined as norm(robot Current Pose (1:2)-Goal Position (:)). where robot Current Pose = [robot Initial Location initial Orientation] and initial Orientation = 0, the desired linear velocity for the robot is 0.5 m/sec, the robot wheel radius is 0.034 m, and the maximum angular velocity is equal to the linear velocity divided by the wheel radius. Given is the optimized function. Eq. (20).

(1) Case 1: The goal is static

In this case, the mean distance for Map 1 is 12.6436 m and the error is 0.0012 m and the elapsed-time is 98.9167 sec; as shown in Fig. 7 and in Table 3.

(2) Case 2: The goal is dynamic

The objective in this scenario is a dynamic that switches between locations at each succeeding time-step interval. The dynamic goal's speed and direction are presumptively random. The configuration is identical to case 1. As shown in Fig. 8 and in Table 4, the mean distance for Map 2 is 12.4961 m, the error is 7.9903×10^{-4} m, and the elapsed-time is 37.5303 sec.

(3) Case 3: dynamic environment

Ten dynamic obstacles were used to test the proposed algorithm in a dynamic environment. The configuration is identical to that of a static environment. The velocity ($vobs$) and direction (θobs) of The dynamic obstacles are regarded as random according to Eq. (23) and (24). As shown in Fig. 9 and in Table 5, the mean distance for Map 3 is 17.6547m, the error is 9.0404×10^{-4} m, and the elapsed-time is 52.2038 sec.

7. Conclusions

The path planning algorithm for mobile robots that was proposed in this paper combined a local search strategy, an obstacle detection strategy, and an improved hunter-prey optimization algorithm. Also, the proposed method is compared to the standard algorithm using 13 benchmark test functions. It considers the actual size of the mobile robot, a kinematic model, and the robot's specifications in an effort to simulate the real world. To reduce the multiple objectives of path length and minimum angles, the algorithm was tested in static and dynamic environments with various scenarios. The simulation results lead to the conclusion that the suggested algorithm has successfully avoided static and moving obstacles with a short execution time. Test functions showed that the suggested algorithm provides significant improvements over five optimization algorithms. The H/W implementation of the suggested algorithm-based path planning on a real Turtlebot robot will be interesting to in the future.

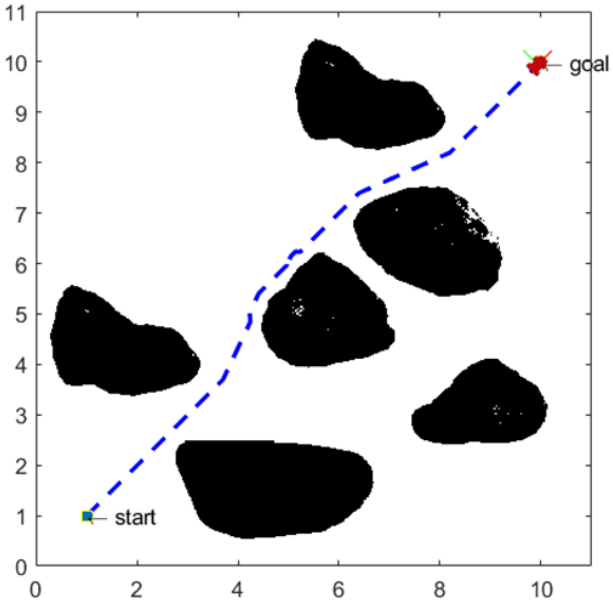

Figure. 7 Map 1

Table 2. Over 1000 times and 30 dimensions were run through the Benchmark test functions

Fun.	Fit.	PSO [28] 30 Dim	SSA [29] 30 Dim	FDO [30] 30 Dim	COOT[31] 30 Dim	HPO [25] 30 Dim	IHPO
F1	min	7.5124×10^{-08}	1.6368×10^{-08}	$3.2270 \times 10^{+03}$	9.8206×10^{-58}	0	0
	max	4.0190×10^{-05}	6.6116×10^{-07}	$1.0719 \times 10^{+04}$	8.6061×10^{-08}	0	0
	avg	2.2965×10^{-06}	1.3651×10^{-07}	$6.3139 \times 10^{+03}$	8.7095×10^{-11}	0	0
	std	7.2554×10^{-06}	1.5491×10^{-07}	$1.8230 \times 10^{+03}$	2.7215×10^{-09}	0	0
F2	min	9.0072×10^{-05}	1.0536×10^{-01}	$2.6008 \times 10^{+01}$	2.9703×10^{-30}	6.2170×10^{-198}	0
	max	8.6400×10^{-03}	$3.1663 \times 10^{+00}$	$4.9082 \times 10^{+01}$	8.5677×10^{-05}	4.0213×10^{-180}	6.6346×10^{-273}
	avg	1.4400×10^{-03}	$1.3328 \times 10^{+00}$	$3.3957 \times 10^{+01}$	3.8193×10^{-07}	5.8110×10^{-183}	6.6346×10^{-276}
	std	1.9200×10^{-03}	9.3288×10^{-01}	$5.7193 \times 10^{+00}$	4.6164×10^{-06}	0	0
F3	min	$2.4899 \times 10^{+01}$	$1.9521 \times 10^{+02}$	$7.1684 \times 10^{+03}$	4.9308×10^{-57}	0	0
	max	$3.6510 \times 10^{+02}$	$2.8737 \times 10^{+03}$	$2.9706 \times 10^{+04}$	1.4499×10^{-07}	1.4560×10^{-290}	0
	avg	$1.2401 \times 10^{+02}$	$1.2921 \times 10^{+03}$	$1.9184 \times 10^{+04}$	1.7311×10^{-10}	1.5673×10^{-293}	0
	std	$8.1552 \times 10^{+01}$	$7.2173 \times 10^{+02}$	$6.0544 \times 10^{+03}$	4.6336×10^{-09}	0	0
F4	min	8.7640×10^{-01}	$3.5141 \times 10^{+00}$	$2.4018 \times 10^{+01}$	1.6363×10^{-34}	4.2538×10^{-169}	0
	max	$4.9311 \times 10^{+00}$	$1.7169 \times 10^{+01}$	$3.9262 \times 10^{+01}$	5.9902×10^{-04}	1.2689×10^{-150}	8.2066×10^{-262}
	avg	$2.0211 \times 10^{+00}$	$9.5648 \times 10^{+00}$	$3.2439 \times 10^{+01}$	6.2591×10^{-07}	1.3881×10^{-153}	8.2066×10^{-265}
	std	8.7088×10^{-01}	$3.1342 \times 10^{+00}$	$3.7864 \times 10^{+00}$	1.8948×10^{-05}	4.0198×10^{-152}	0
F5	min	$1.2929 \times 10^{+01}$	$2.0042 \times 10^{+01}$	$2.0875 \times 10^{+02}$	$2.6813 \times 10^{+01}$	20.6792	4.9762×10^{-16}
	max	$8.4144 \times 10^{+01}$	$1.2065 \times 10^{+03}$	$9.4832 \times 10^{+06}$	$1.6774 \times 10^{+03}$	25.9952	23.5571
	avg	$3.0213 \times 10^{+01}$	$2.0168 \times 10^{+02}$	$4.9340 \times 10^{+06}$	$4.5168 \times 10^{+01}$	21.8228	0.4859
	std	$1.9874 \times 10^{+01}$	$2.7394 \times 10^{+02}$	$1.8511 \times 10^{+06}$	$7.1811 \times 10^{+01}$	0.4880	2.5575
F6	min	3.1085×10^{-08}	2.2926×10^{-08}	$2.9047 \times 10^{+03}$	1.2200×10^{-02}	2.2110×10^{-14}	0
	max	7.8797×10^{-06}	6.5254×10^{-07}	$1.1198 \times 10^{+04}$	$1.2563 \times 10^{+00}$	5.9938×10^{-09}	5.6221×10^{-12}
	avg	1.2411×10^{-06}	1.6204×10^{-07}	$6.0501 \times 10^{+03}$	1.4580×10^{-01}	1.5478×10^{-11}	5.6221×10^{-15}
	std	1.9515×10^{-06}	1.5808×10^{-07}	$1.7282 \times 10^{+03}$	1.1980×10^{-01}	2.1548×10^{-10}	1.7779×10^{-13}
F7	min	8.1600×10^{-03}	4.9200×10^{-02}	$1.0362 \times 10^{+00}$	5.3725×10^{-05}	2.3086×10^{-07}	2.1034×10^{-08}
	max	4.4880×10^{-02}	3.9160×10^{-01}	$6.5583 \times 10^{+00}$	4.2500×10^{-02}	0.0010	6.6430×10^{-05}
	avg	2.0240×10^{-02}	1.3672×10^{-01}	$2.7259 \times 10^{+00}$	5.1000×10^{-03}	1.0961×10^{-04}	7.8913×10^{-07}
	std	8.2400×10^{-03}	7.2800×10^{-02}	$1.2317 \times 10^{+00}$	4.3000×10^{-03}	1.3757×10^{-04}	5.4790×10^{-06}
F8	min	$-6.3116\times 10^{+03}$	$-7.1509 \times 10^{+03}$	$-3.2318 \times 10^{+03}$	$-1.2185 \times 10^{+04}$	$-1.1561 \times 10^{+04}$	$-8.7196 \times 10^{+03}$
	max	$-3.8802 \times 10^{+03}$	$-4.8477 \times 10^{+03}$	$-1.7393 \times 10^{+03}$	$-4.9284 \times 10^{+03}$	$-7.7201 \times 10^{+03}$	$-7.1753 \times 10^{+03}$
	avg	$-5.2250 \times 10^{+03}$	$-5.9466 \times 10^{+03}$	$-2.2258 \times 10^{+03}$	$-7.3315 \times 10^{+03}$	$-9.3622 \times 10^{+03}$	$-8.4011 \times 10^{+03}$
	std	$6.4182 \times 10^{+02}$	$6.1953 \times 10^{+02}$	$3.2376 \times 10^{+02}$	$9.1334 \times 10^{+02}$	573.0050	387.4515
F9	min	$7.6413 \times 10^{+01}$	$2.3083 \times 10^{+01}$	$1.2846 \times 10^{+02}$	$0.0000 \times 10^{+00}$	0	0
	max	$7.6413 \times 10^{+01}$	$7.9597 \times 10^{+01}$	$1.8942 \times 10^{+02}$	3.4416×10^{-06}	0	0
	avg	$3.8604 \times 10^{+01}$	$4.7476 \times 10^{+01}$	$1.6083 \times 10^{+02}$	3.4531×10^{-09}	0	0
	std	$1.2983 \times 10^{+01}$	$1.4882 \times 10^{+01}$	$1.5238 \times 10^{+01}$	1.0883×10^{-07}	0	0
F10	min	3.6847×10^{-05}	$1.3170 \times 10^{+00}$	$8.1832 \times 10^{+00}$	8.8818×10^{-16}	8.8818×10^{-16}	8.8818×10^{-16}
	max	$2.2509 \times 10^{+00}$	$3.5064 \times 10^{+00}$	$1.1314 \times 10^{+01}$	2.4877×10^{-05}	8.8818×10^{-16}	8.8818×10^{-16}
	avg	$1.0174 \times 10^{+00}$	$2.0904 \times 10^{+00}$	$1.0321 \times 10^{+01}$	7.3884×10^{-08}	8.8818×10^{-16}	8.8818×10^{-16}
	std	6.5720×10^{-01}	5.0912×10^{-01}	7.7416×10^{-01}	1.0913×10^{-06}	0	0
F11	min	3.0870×10^{-08}	7.1483×10^{-04}	$2.7147 \times 10^{+01}$	$0.0000 \times 10^{+00}$	0	0
	max	5.6720×10^{-02}	3.6400×10^{-02}	$8.6144 \times 10^{+01}$	4.3979×10^{-07}	0	0
	avg	1.1840×10^{-02}	1.3520×10^{-02}	$5.5598 \times 10^{+01}$	4.3990×10^{-10}	0	0
	std	1.3360×10^{-02}	9.3600×10^{-03}	$1.5810 \times 10^{+01}$	1.3907×10^{-08}	0	0
F12	min	2.4686×10^{-09}	$2.2231 \times 10^{+00}$	$5.4161 \times 10^{+04}$	5.3513×10^{-04}	1.6046×10^{-15}	1.5705×10^{-32}
	max	7.4704×10^{-01}	$8.9368 \times 10^{+00}$	$8.2632 \times 10^{+06}$	$4.5544 \times 10^{+00}$	0.0065	7.5616×10^{-07}
	avg	1.4120×10^{-01}	$4.8889 \times 10^{+00}$	$2.8270 \times 10^{+06}$	2.0060×10^{-01}	6.5451×10^{-06}	7.5616×10^{-10}
	std	1.9880×10^{-01}	$1.9097 \times 10^{+00}$	$1.9333 \times 10^{+06}$	4.8000×10^{-01}	2.0697×10^{-04}	2.3912×10^{-08}
F13	min	2.7983×10^{-07}	7.2400×10^{-02}	$3.2630 \times 10^{+06}$	2.1300×10^{-02}	7.8331×10^{-14}	0.0974
	max	4.9800×10^{-01}	$3.4803 \times 10^{+01}$	$2.2232 \times 10^{+07}$	$4.1765 \times 10^{+00}$	1.1848	1.3136
	avg	2.7680×10^{-02}	$1.3999 \times 10^{+01}$	$1.0170 \times 10^{+07}$	4.5590×10^{-01}	0.1053	0.1109
	std	9.1360×10^{-02}	$1.1228 \times 10^{+01}$	$5.3626 \times 10^{+06}$	4.7000×10^{-01}	0.1303	0.1240

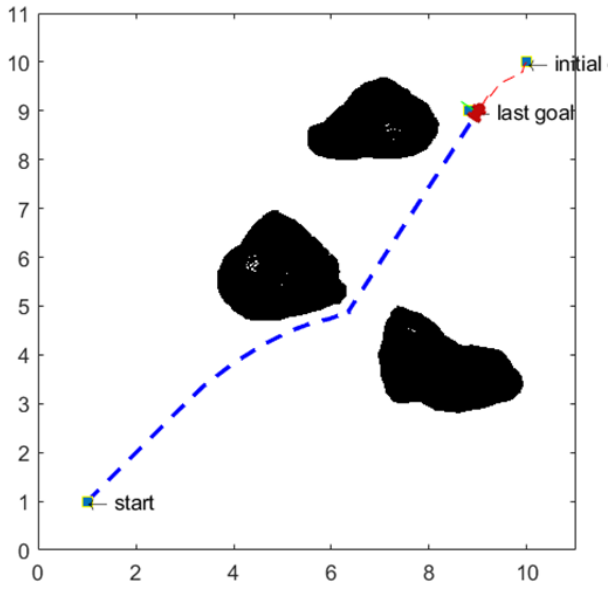

Figure. 8 Map 2

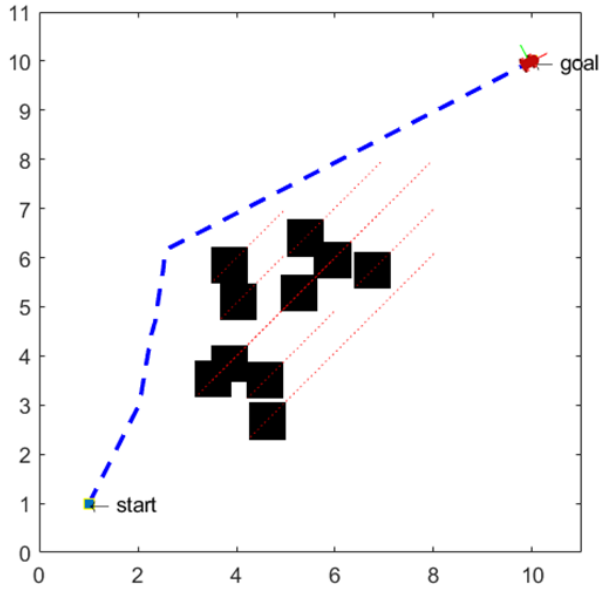

Figure. 9 Map 3

Table 3. Case 1 perfromance

Run	Distance (m)	Error (m)	Time (sec)
1	12.6605	0.0013	68.1097
2	12.7279	0.0010	89.8222
3	12.7279	0.0012	117.3200
4	12.6606	0.0014	99.7441
5	12.7279	0.0010	98.0004
6	12.4366	0.0012	101.1168
7	12.6836	0.0013	93.6202
8	12.4591	0.0013	104.4598
9	12.7279	0.0012	112.0621
10	12.6235	0.0012	104.9120
Mean	**12.6436**	**0.0012**	**98.9167**

Table 4. Case 2 perfromance

Run	Distance (m)	Error (m)	time (sec)
1	10.7438	7.8921×10^{-4}	45.9580
2	14.3865	9.7320×10^{-4}	30.0764
3	10.9070	5.7674×10^{-4}	34.7944
4	13.3758	8.6215×10^{-4}	37.5094
5	12.3599	8.6533×10^{-4}	48.8956
6	13.7566	8.7984×10^{-4}	34.2336
7	13.5236	8.8732×10^{-4}	43.2677
8	11.8127	9.0469×10^{-4}	34.7869
9	10.8405	5.3174×10^{-4}	32.7131
10	13.2550	7.2011×10^{-4}	33.0675
Mean	**12.4961**	**7.9903×10^{-4}**	**37.5303**

Table 5. Case 3 perfromance

Run	Distance (m)	Error (m)	Time (sec)
1	15.8410	7.6359×10^{-4}	59.2166
2	16.8796	6.7539×10^{-4}	64.7438
3	17.2434	8.8301×10^{-4}	44.9348
4	22.8446	0.0014	75.3341
5	17.5226	9.1849×10^{-4}	50.2706
6	14.7211	5.8216×10^{-4}	53.6833
7	18.1267	8.8401×10^{-4}	43.5966
8	18.2776	6.3378×10^{-4}	44.2049
9	15.6198	0.0011	41.4114
10	19.4711	0.0012	44.6419
Mean	**17.6547**	**9.0404×10^{-4}**	**52.2038**

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Jaafar Ahmed Abdulsaheb PhD candidate contributed to methodology, classification model proposed, software, and writing review and editing. Dheyaa Jasim Kadhim contributed to supervising the overall work and editing the paper.

References

[1] B. K. Patle, G. L. Babu, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot", *Defence Technology*, Vol. 15, No. 4, pp. 582–606, Aug. 2019, doi: 10.1016/j.dt.2019.04.011.

[2] P. Ehlert, "The use of Artificial Intelligence in autonomous mobile robots", *Report on Research Project, Delft University of Technology*, Netherlands, 1999.

[3] J. M. Keil, "Decomposing a Polygon into Simpler Components", *SIAM Journal on*

Computing, Vol. 14, No. 4, pp. 799–817, 1985, doi: 10.1137/0214056.

[4] C. Zhong, S. Liu, B. Zhang, Q. Lu, J. Wang, Q. Wu, and F. Gao, "A Fast On-line Global Path Planning Algorithm Based on Regionalized Roadmap for Robot Navigation", *IFAC-PapersOnLine*, Vol. 50, No. 1, pp. 319–324, Jul. 2017, doi: 10.1016/j.ifacol.2017.08.053.

[5] U. O. Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field", *Appl Soft Comput*, Vol. 77, pp. 236–251, 2019, doi: 10.1016/j.asoc.2019.01.036.

[6] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "Heuristic approaches in robot path planning: A survey", *Rob Auton Syst*, Vol. 86, pp. 13–28, 2016, doi: 10.1016/j.robot.2016.08.001.

[7] J. Ou and M. Wang, "Path Planning for Omnidirectional Wheeled Mobile Robot by Improved Ant Colony Optimization", In: *Proc. of Chinese Control Conf., CCC, IEEE Computer Society, Guangzhou, China*, pp. 2668–2673, Jul. 2019, doi: 10.23919/ChiCC.2019.8866228.

[8] 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.

[9] 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", In: *12th World Congress on Intelligent Control and Automation (WCICA), Guilin, China*, pp. 1496–1501, 2016, doi: 10.1109/WCICA.2016.7578817.

[10] 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.

[11] P. K. Das, S. K. Pradhan, S. N. Patro, and B. K. Balabantaray, "Artificial Immune System Based Path Planning of Mobile Robot", *Studies in Computational Intelligence*, pp. 195–207, 2012, doi: 10.1007/978-3-642-25507-6_17.

[12] T. K. Dao, T. S. Pan, and J. S. Pan, "A multi-objective optimal mobile robot path planning based on whale optimization algorithm", In: *Proc. of IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China*, pp. 337–342, 2016, doi: 10.1109/ICSP.2016.7877851.

[13] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning", *Procedia Comput Sci*, Vol. 127, pp. 180–189, 2018, doi: 10.1016/j.procs.2018.01.113.

[14] D. Davis and P. Supriya, "Implementation of Fuzzy-Based Robotic Path Planning", in *Advances in Intelligent Systems and Computing*, pp. 375–383, 2016, doi: 10.1007/978-81-322-2523-2_36.

[15] 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.

[16] X. Liang, D. Kou, and L. Wen, "An Improved Chicken Swarm Optimization Algorithm and its Application in Robot Path Planning", *IEEE Access*, Vol. 8, pp. 49543–49550, 2020, doi: 10.1109/ACCESS.2020.2974498.

[17] F. Li, X. Fan, and Z. Hou, "A Firefly Algorithm With Self-Adaptive Population Size for Global Path Planning of Mobile Robot", *IEEE Access*, Vol. 8, pp. 168951–168964, 2020, doi: 10.1109/ACCESS.2020.3023999.

[18] Y. Quan, H. Ouyang, C. Zhang, S. Li, and L. Q. Gao, "Mobile Robot Dynamic Path Planning Based on Self-Adaptive Harmony Search Algorithm and Morphin Algorithm", *IEEE Access*, Vol. 9, pp. 102758–102769, 2021, doi: 10.1109/ACCESS.2021.3098706.

[19] L. Shao, Q. Li, C. Li, and W. Sun, "Mobile Robot Path Planning Based on Improved Ant Colony Fusion Dynamic Window Approach", In: *Proc. of IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan*, pp. 1100–1105, Aug. 2021, doi: 10.1109/ICMA52036.2021.9512795.

[20] P. Kusuma and A. Prasasti, "Guided Pelican Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, pp. 179–190, 2022, doi: 10.22266/ijies2022.1231.18.

[21] P. Kusuma and M. Kallista, "Stochastic Komodo Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, 2022, doi: 10.22266/ijies2022.0831.15.

[22] P. Kusuma and A. Dinimaharawati, "Fixed Step Average and Subtraction Based Optimizer", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, 2022, doi: 10.22266/ijies2022.0831.31.

[23] F. Zeidabadi and M. Dehghani, "POA: Puzzle Optimization Algorithm", *International Journal*

of Intelligent Engineering and Systems, Vol. 15, No. 1, 2022, doi: 10.22266/ijies2022.0228.25.

[24] N. Abbas and J. Abdulsaheb, "An Adaptive Multi-Objective Particle Swarm Optimization Algorithm for Multi-Robot Path Planning", *Journal of Engineering (Eng. J.)*, Vol. 22, No. 7, pp. 164–181, 2016.

[25] I. Naruei, F. Keynia, and A. S. Molahosseini, "Hunter–prey optimization: algorithm and applications", *Soft comput*, Vol. 26, No. 3, pp. 1279–1314, 2022, doi: 10.1007/s00500-021-06401-0.

[26] W. A. Mahmoud and D. J. Kadhim, "A Proposal Algorithm to Solve Delay Constraint Least Cost Optimization Problem", *Journal of Engineering*, Vol. 19, No. 1, pp. 155–160, 2013.

[27] J. A. Abdulsaheb and D. J. Kadhim, "Robot Path Planning in Unknown Environments with Multi-Objectives Using an Improved COOT Optimization Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 5, pp. 548–565, 2022, doi: 10.22266/ijies2022.1031.48.

[28] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, doi: 10.1109/MHS.1995.494215.

[29] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems", *Advances in Engineering Software*, Vol. 114, pp. 163–191, Dec. 2017, doi: 10.1016/j.advengsoft.2017.07.002.

[30] J. M. Abdullah and T. Ahmed, "Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process", *IEEE Access*, Vol. 7, pp. 43473–43486, 2019, doi: 10.1109/ACCESS.2019.2907012.

[31] I. Naruei and F. Keynia, "A new optimization method based on COOT bird natural life model", *Expert Syst Appl*, Vol. 183, p. 115352, 2021, doi: 10.1016/j.eswa.2021.115352.