Research Article

Robot path planning based on laser range finder and novel objective functions in grey wolf optimizer

Navid Toufan[1] · Aliakbar Niknafs[2]

Received: 16 March 2020 / Accepted: 16 June 2020 / Published online: 3 July 2020

Abstract
Mobile robots are the robots that can move through the environment and be used in many applications, including the industrial environment, planet exploration, warehousing, and daily household chores. They can be controlled by an operator, set to do some specific jobs, or work autonomously. Robot path planning is the task of an autonomous robot to move safely from one position to another. In this paper, three new objective functions are introduced in the structure of improved grey wolf optimizer (IGWO) and improved particle swarm optimization (IPSO) for the robot path planning problems. As another part of our proposed method, a reduction of laser range finder (LRF) data is performed, and the avoidance collision approach is also introduced. Robots determine the next position by using LRF data and IGWO (IPSO) algorithms in a local approach. The initial and the goal positions are predefined for each robot. Moreover, the location of static obstacles and other robots are unknown for each robot. Finally, the experimental results of the robot path planning using IGWO are compared to different algorithms. The results indicate that the proposed method performs better in determining an optimal, short, safe, and smooth path. Also, it has less power and time consumption than other methods. All the algorithms are implemented in the V-REP robot simulator.

Keywords Multi robot path planning · Grey wolf optimizer · Multi-objective function · Optimal path · Laser range finder · V-REP robot simulator

1 Introduction

The role of path planning is acquiring a safe path by a robot in an environment from a predefined initial position to a target position with respect to optimal path and local constraints [36]. During the last decade, the field of path planning has been heavily studied in both academics and industry due to its applications in many real tasks like manufacturing, automobile, medical, industries, and so on. In the case of map representation, two discrete and continuous spaces could be considered based on environment information. The environment complexity changes under the influence of static and dynamic obstacles and even the presence of other robots on a common map. The strategy in which the robots achieve the goal based on environment information could be categorized into two local and global path planning problems [5]. In the global path planning [55], which is known as offline planning, the robots start moving after determining a collision-free trajectory from an initial position to a goal position. While in the local path planning or online planning [46], the robots navigate through the map step by step and find the next position toward the target.

The multi-robot path planning problem could be classified into two main approaches: centralized and distributed. In the centralized approach [27, 48], all the individual

✉ Aliakbar Niknafs, niknafs@uk.ac.ir; Navid Toufan, n.toofan@eng.uk.ac.ir; navid.toofan@gmail.com | [1]Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran. [2]Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.

SN Applied Sciences (2020) 2:1324 | https://doi.org/10.1007/s42452-020-3093-5

robots are considered in a large multiplex system and the robot's objective functions and the constraints are computed through a path planning algorithm. Although this approach guarantees the completeness, high time complexity might occur as a result of increasing the dimensionality of configuration space. On the other hand, in the distributed or decoupled approach [12, 31], each robot plans the path independently. Actually, in this approach, each robot employs its algorithm, leading to faster convergence and reduced execution time. The disadvantage of the distributed approach appears to be the failure of coordination between the generated path and the collision-free trajectory of all the robots.

Some various methods have been implemented on the motion or path planning problems. Approaches like roadmap, cell decomposition, potential field, and mathematical programming are classified in the classical approach. In the roadmap approach, the free C-space and the set of conceivable motions are reduced to the start and goal points in the C-Space that could be connected by a path. The two well-known roadmaps are visibility graph [1] and Voronoi diagram [4]. Šeda [45] compared the roadmap and cell decomposition approaches with each other. The potential field approach [3] uses the combination results of attraction and repulsion forces, and conducts the robot toward the target. The mathematical programming approach acts toward the trajectory planning problem as a numerical optimization problem such as MILP method [44]. Since the classical approach was incapable in some situations such as high dimensionality, time complexity, and trapping in local minima, the probabilistic approach has been developed. The probabilistic road maps (PRM) and rapidly-exploring random trees (RRT) are the most influential sampling-based motion planning algorithms which perform well in the high-dimensional state spaces [24].

Due to the above drawbacks of classical approaches, the other approaches like the heuristic approach have been developed in the field of artificial intelligence to fix the drawbacks. The heuristic and metaheuristic algorithms provide better performance because of their inherent features. Since the metaheuristic algorithms incorporate the local search and randomization method, they offer better equivalence among the diversification and intensification searches for the local and global optimums. The solution is detected much faster in heuristic algorithms compared to deterministic methods, but they do not guarantee to find it.

The metaheuristic algorithms are divided into two categories: individual solution algorithms and population based algorithms. The individual solution algorithms generate a random solution and improve the single solution until it reaches the optimum result. Tabu search (TS) and simulated annealing (SA) approaches are the well-known algorithms applied in different studies to solve the optimization problems. TS algorithm is used to find an optimal path in [29] for the first time. Nevertheless, the large number of iterations and too many parameters for adjustment can be considered as the drawbacks of TS algorithm. In [32], a SA approach is proposed to achieve an optimal or near optimal path for the mobile robot in the environments that include both static and dynamic obstacles. However, the algorithm suffers from slow convergence speed and long execution time, and its performance relies on the initial value.

On the other hand, in population based algorithms, a set of solutions are stochastically generated in a defined search space and the solutions will be updated in each iteration until the best value is achieved. The evolutionary and swarm intelligence are the metaheuristic algorithms which can solve the multi-objective optimization problems. Genetic algorithm (GA) is a famous optimization algorithm based on natural genetic that takes merits from the mechanisms such as natural selection, crossover, and mutation. It has the strong search capability and high search efficiency. In [37], the idea of using GA in robot path planning problem is introduced. Hocaoglu and Sanderson [21] proposed a novel iterative multi-resolution path planning method on the basis of GA. In some cases, the combination of GA and other algorithms provide better results. In [35], GA is used to optimize the trajectory with the assistance of the Neural Network approach, and it provides better learning in complex systems. However, it tends to premature convergence.

Particle swarm optimization (PSO) is one of the popular optimization algorithms because of its simple implementation, fast convergence, and short computational complexity. In [39], the path planning is done in three phases: first, the MAKLINK graph is used to build the search space of the mobile robot. Then a Dijkstra algorithm provides the shortest path from the initial state to the goal state. Finally, the optimal path is achieved by adding a mutation operator to the PSO algorithm. The mutation operator keeps the fast speed characteristic of PSO in earlier phases and also provides the local minima avoidance ability for PSO in the last phases. The results demonstrate the effectiveness of the proposed method in the mobile robot navigation problem. Wang et al. [49] developed an obstacle avoidance path planning approach in the field of soccer robot by defining an effective fitness function in PSO. The results indicate that this approach could generate a high quality solution in a reasonable time with the low complexity and rapid convergence. Also, in [28], the path planning in an unknown environment is extended by the particle swarm intelligence, and the fitness of particles is evaluated based on the positions of

goals and obstacles in the environment. The global best position in each iteration is achieved by the robot until it arrives to the goal. The simulation shows that the robot reaches its target without the collision. Nasrollahy and Javadi [34] attempted to find a collision-free path from the start status to the moving target in a dynamic environment in which both static and dynamic obstacles are present. The proposed method reduces the total mobile robot path planning time and avoids the local optimum trapping.

The hybridization of PSO with the other algorithms helps to find a better solution in the search space. The combination of PSO and PRM has been developed in [30], in which PSO and PRM have been used for the global path planning and the collision-free path, respectively. In addition, two objective functions have been reserved in the PSO algorithm to provide a short and smooth path from an initial position to the final position. The proposed method is compared to RPM method and the results show that the proposed approach could find a shorter and smoother path in a less time. In [47] a new particle swarm optimization algorithm is introduced to track the pedestrians in a crowded scene. The social interaction between the swarm and output of pedestrians detection are incorporated in the velocity-updating equation to track the multiple targets in a crowded scene. The experimental results show that the proposed method preforms better than other methods with a high accuracy. In [7], a hybrid IPSO-IGSA algorithm is introduced by the combination of improved PSO and improved GSA that could provide an effectual balance between the exploration and exploitation using the improved GSA acceleration and improved PSO velocity to update the particle positions. Also, Ju et al. [23] proposed a hybrid PSO-GA algorithm using a scalable encoding technique for the path planning problems. The results indicate that the proposed hybrid algorithm causes fewer turning point generation in case of finding a shorter collision-free path.

Ant colony optimization (ACO) is another swarm intelligence algorithm inspired from the natural ant colony behavior. At the essence of the behavior, the interacted communication between the ants enables them to discover the shortest route between their nest and the food sources. ACO has been used in [41] to decrease the path length in a traffic situation. Although the ant colony optimization guarantees to converge, the convergence time is uncertain. The other algorithms such as artificial bee colony (ABC) optimization have been combined with the chaos search treatment to propose a new approach in robot path planning problems. The method which has been proposed in [22], introduces a new local path planning approach in a dynamic environment to find a possible shortest path by using two cost functions in bacterial foraging optimization (BFO).

Ant lion optimization (ALO) algorithm is designed based on hunting the procedure of ant lions. The exploration of ALO includes the random walk and random selection of agents. Also, the exploitation is done with traps. A multi-objective ant lion optimization approach has been utilized in [8] to reach an optimal trajectory with minimization time-jerk-torque for a 6-axis manipulator industrial robot. Whale optimization algorithm (WOA) is a new swarm intelligence algorithm that is inspired by the humpback whales. Although it avoids the local optima and reaches a global optimal solution, it has low convergence precision and convergence rate. In [6], an improved whale optimization algorithm is applied to find a collision-free route. The proposed method is tested in different search spaces. Zheng [54] introduced water wave optimization (WWO) method, where the search space imitates the seabed area and each solution is similar to a "wave" with a height h and a wavelength λ, and the solution proficiency is measured by its seabed depth: the shorter the distance of the current water level, the higher the fitness is. Some advantages of WWO include simple algorithmic framework, easy implementation, and few control parameters. In [52], a hybrid algorithm based on WWO and neighbourhood search is introduced in order to solve quarantine vehicle scheduling problem for high-risk isolated transfers. The computational results illustrate that the proposed algorithm considerably outperforms several famous algorithms and obtains high-quality solutions on a real-world problem. However, WWO falls easily into the local optimum, and has low calculation accuracy.

Grasshopper optimization algorithm (GOA) is introduced by Saremi et al. [43] to mimic the behavior of grasshopper swarms in the nature for solving optimization problems. Based on the mathematical model introduced for GOA, the movement of grasshopper is substantially influenced by the following factors: social interaction, gravity force and wind advection. The implementation of GOA is simple, and it has high accuracy. In [50], a dynamic GOA is proposed for optimizing the distributed trajectory of UAVs in urban environments. The results demonstrate that GOA algorithm can obtain an enhanced performance and satisfactory trajectories. The main drawbacks of GOA are slow convergence speed and unbalanced exploration–exploitation ability. In 2019, a new nature-inspired algorithm called Harris Hawks optimizer (HHO) is established by Heidari et al. [20]. The idea of HHO is stimulated from the cooperative behaviors of Harris's hawks to trap the escaping preys. Similar to other metaheusristic algorithms, HHO tends to get stuck in local optima and has an unbalanced exploitation ability.

Grey wolf optimizer (GWO) algorithm has been introduced by Mirjalili et al. [33] as a developed optimization approach that is very well-organized to

approximate the optimum of the optimization problem. Zhang et al. [53] used grey wolf optimizer (GWO) in the field of unmanned combat aerial vehicle (UCAV) two-dimensional path planning problem to find an optimal flight route with the constraints. However, homologous with the other metaheuristic search algorithms, GWO also confronts the problem of getting trapped in sub-optimal solutions and premature convergence in several cases. The search ability of GWO is improved in [16] by introducing RW-GWO based on random walk. It is argued that the novel algorithm has a significant performance to solve both continuous and real life optimization problems rather than the classical GWO. In [19], RW-GWO is utilized to solve the complex, nonlinear and constrained reliability problems. The results conclude that RW-GWO has a better performance for solving the complex problems compared to GWO and other algorithms. Also, in [18], RW-GWO is applied for directional overcurrent relay problems in order to find an optimal setting.

In [17], an improved version of GWO is introduced by using OBL and chaotic local search (OCS-GWO) to prevent the algorithm from getting stuck in the local optima. The OCS-GWO provides a good balance between the exploration and exploitation. The results indicate better performance of the proposed algorithm compared to traditional GWO. In some complex real-world non-linear optimizations, the performance of GWO is not satisfactory due to insufficient search ability. Therefore, a Cauchy operator has been utilized to increase the exploration ability along with the exploitation in [13]. In the modified Cauchy-GWO algorithm, two new wolves are generated through Cauchy distributed random numbers and the next new wolf is produced by taking the convex integration of those new wolves. The performance of Cauchy-GWO has been evaluated on CEC 2014 test function which the outcomes indicate the superiority of Cauchy-GWO over the original GWO. The hybridization of Levy-flight search mechanism and traditional GWO (GLF-GWO) is introduced in [14] in order to improve the search efficiency of the leading hunter in the original GWO and enhance the search capability of GWO through providing the appropriate guidance. The results show that the performance of the proposed GLF-GWO is more better than classical GWO. Also, Gupta and Deep [15] introduced two new algorithms by hybridizing GWO with DE mutation namely DE-GWO and gDE-GWO in order to avoid the stagnation of the solution in GWO. The results illustrate that the proposed gDE-GWO algorithm has a considerable robustness to solve a set of 32 well-known benchmark test problems compared to GWO and DE-GWO.

The other AI algorithms such as fuzzy logic, neural network, and reinforcement learning could be used in path planning problems alone or alongside the other classical and modern approaches. In [42, 51], a combination of fuzzy logic and neural network is used in the path planning problem to find an optimal solution. Frontzek et al. [10] introduced a flexible motion planning approach for the real-time applications by exploiting A* method and neural radial basis function networks (RBFN). Reinforcement learning (RL) enables a robot to autonomically discover an optimal behavior through trial-and-error interactions with its environment. In RL, the expert of a control task provides a feedback in terms of a scalar objective function [25]. Q-Learning is a value-based reinforcement learning algorithm. In [26], a new deterministic Q-learning method stores the Q-value for the best possible action at a state with the reduced time and storage.

2 Basic concepts

2.1 Particle swarm optimization (PSO)

The nature-inspired algorithms are derived from the communal behavior of different species such as birds, fishes, insects, etc. Particle swarm optimization (PSO) has been inspired by a flock of birds by Eberhart and Kennedy [9]. PSO is a computational technique that optimizes a problem by iteratively attempting to improve a candidate solution with consideration to a given measure of quality. It solves a problem by distributing a population of candidate solutions (particles). These particles are moved through the search space toward an optimal solution by iteratively updating the position and velocity parameters via a simple mathematical formula. The particles in the search space share the information between different individuals to update the velocity and subsequently the position in order to detect the best position entire the search space. Easy implementation, efficient computational time, and adjustment of a few parameters make PSO suitable to be used in different applications. During the last decade, some of the enhanced PSO methods such as improved BPSO [38] and area extension PSO [2] have been proposed to provide a good balance between the exploitation and exploration by changing the weight value and the acceleration coefficients during each iteration. A modified IPSO equation which has been introduced in [7], is presented in Eqs. (1) and (2).

$$V_i^d(t+1) = W_i V_i^d(t) + C_1 r_1 \left(X^d_{Pbest_i} - X_i^d(t) \right) + C_2 r_2 \left(X^d_{Gbest} - X_i^d(t) \right) \quad (1)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1). \quad (2)$$

Suppose N is the population of particles, each particle is defined by two characteristics: position and velocity. $X_i = (x_{i1}, x_{i2}, \ldots x_{iD})$ represents the current position vector of *i*th particle in a D-dimensional search space and the velocity of *i*th particle is indicated by $V_i = (v_{i1}, v_{i2}, \ldots v_{iD})$.

$P_i best$ and *Gbest* are represented in Eqs. (3) and (4) as follows:

$$P_i best(t+1) = \begin{cases} P_i best(t) & \text{if } Obj(X_i(t+1) > Obj(P_i best(t)) \\ X_i(t+1) & Otherwise \end{cases} \tag{3}$$

$$Gbest(t+1) = Argmin\{Obj(P_1 best(t+1)), Obj(P_2 best(t+1)) , \ldots Obj(P_N best(t+1))\}. \tag{4}$$

In each iteration which is indicated by t, each particle is updated by the following two best values. The first one is the individually best solution that has achieved so far, which is called *Pbest*. Another best value which has obtained so far by any particle in the population, is tracked by the particle swarm optimizer and called *Gbest*.

For enhanced understanding, note that Eq. (1) is composed of three parts. In the first part, $W_i V_i^d(t)$, the inertial weight which is referred to W, provides the searchability in the PSO algorithm. When W is large, it provides the global search capability, and when it is small, it makes the local searchability for the algorithm. So the dynamic change of inertia weight adjusts the searchability dynamically. The inertial weight proposed in [7] is represented in Eq. (5).

$$W_i(t) = W_{min} + (W_{max} - W_{min}) \frac{Dist_i(t)}{\text{Max\_Dist(t)}} \tag{5}$$

where W_{min} and W_{max} are the initial and final inertia weights, respectively. Also, $Dist_i$ is the current Euclidean distance of *i*th particle from the global best which is formulated in Eq. (6), and Max_Dist is the maximum Euclidean distance which is obtained by calculating the distance of all particles from the global best and selecting the largest value among them. It is defined as follows:

$$Dist_i(t) = \sqrt{\sum_{d=1}^{D} \left(X_{Gbest}^d(t) - X_i^d(t)\right)^2} \tag{6}$$

$$\text{Max\_Dist(t)} = Argmax\{Dist_1(t), Dist_2(t), \ldots Dist_N(t)\}. \tag{7}$$

As mentioned above, the two other parts in Eq. (1) are $C_1 r_1 (X_{Pbest_i}^d - X_i^d(t))$ and $C_2 r_2 (X_{Gbest}^d - X_i^d(t))$ which are referred to the cognitive and social components. Generally, in the population-based optimization techniques, it is crucial to improve the convergence toward the global optima in order to find the optimum solution during the last iteration. The time varying acceleration coefficients C_1 and C_2 in Eqs. (8) and (9) have been proposed in [40] to change the searching strategy during the optimization.

$$C_1 = (C_{1f} - C_{1i}) \frac{t}{\text{MaxIter}} + C_{1i} \tag{8}$$

$$C_2 = (C_{2f} - C_{2i}) \frac{t}{\text{MaxIter}} + C_{2i}. \tag{9}$$

A large cognitive component and small social component at the early iteration enable the particle to move through the search space entirely. However, the particles are allowed to converge to the global optima during the optimization by having a small cognitive component and large social component at the last iteration. It is notable that C_{1i}, C_{1f}, C_{2i} and C_{2f} are the fixed values, and MaxIter is the maximum number of iterations.

2.2 Grey wolf optimizer (GWO)

Grey wolves organize themselves into a pack to maintain stability and cooperation in a hunting situation. Grey wolf optimizer is a new metaheuristic optimization method, which has been introduced by Mirjalili et al. [33], and has been inspired from the grey wolves hunting strategies. Firstly, the candidate solutions are generated randomly in the search space in GWO same as the other metaheuristic algorithms. The first three main wolves are called alpha (α), beta (β), and delta (δ). Alpha (α) indicates the first best candidate solution. The second and third best candidate solutions are represented by beta (β) and delta (δ). The other wolves are considered as omega (ω) and they follow α, β, and δ by encircling them. Generally, hunting the prey is done by the wolves in four steps, including encircling pray, hunting, attacking prey, and searching for prey. Equations (10) and (11) are proposed to mathematically simulate the surrounding prey by the wolves.

$$\vec{D} = \left|\vec{C}.\vec{X}_p(t) - \vec{X}(t)\right| \tag{10}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \tag{11}$$

where t represents the present iteration, and $\vec{A}$ and $\vec{C}$ are the coefficient vectors. The position of prey is indicated by $\vec{X}_p(t)$. Also, $\vec{X}$ represents the position of a grey wolf.

The calculation of $\vec{A}$ and $\vec{C}$ coefficient vectors is realized in Eqs. (12) and (13), respectively.

$$\overrightarrow{A(t)} = 2.\overrightarrow{a(t)}.\overrightarrow{r_1} - \overrightarrow{a(t)} \tag{12}$$

$$\overrightarrow{C(t)} = 2.\overrightarrow{r_2} \tag{13}$$

where $\vec{r_1}$ and $\vec{r_2}$ are the random vectors in [0, 1]. The value of $\vec{a}$ parameter is linearly decreased through the course of iteration from 2 to 0 by the Eq. (14).

$$\vec{a} = 2 - \frac{2 * I}{Maxiter}. \tag{14}$$

In the hunting situation, the grey wolves' innate ability helps them to find out prey and encircle it. There is no information about the location of the optimum (prey) in the search space. Therefore, alpha (the best candidate solution), beta, and delta have more wisdom about the prey location to simulate the hunting behavior of grey wolves in the mathematical formulas. So the other wolves update their positions based on the position of alpha, beta and delta wolves as follows:

$$\begin{aligned} &\vec{D}_\alpha = \left|\vec{C_1}\vec{X}_\alpha - \vec{X}(t)\right|, \vec{D}_\beta = \left|\vec{C_2}\vec{X}_\beta - \vec{X}(t)\right| \\ &\vec{D}_\delta = \left|\vec{C_3}\vec{X}_\delta - \vec{X}(t)\right| \end{aligned} \tag{15}$$

$$\vec{X_1} = \vec{X}_\alpha - \vec{A_1},\ \vec{X_2} = \vec{X}_\beta - \vec{A_2},\ \vec{X_3} = \vec{X}_\delta - \vec{A_3} \tag{16}$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{17}$$

where $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$ indicate the positions of alpha, beta and delta, respectively. $\vec{C_1}$, $\vec{C_2}$, $\vec{C_3}$ are the random vectors and $\vec{X}$ denotes the position of current agent.

Attacking prey could be considered as the exploitation, and the search for prey indicates the exploration in the GWO algorithm. The parameter $\vec{A}$ is in the interval $[-a, a]$. When $|A| > 1$ in the first half period of iterations, the algorithm starts to do exploration by decreasing $\vec{a}$ during various iterations and when $|A| < 1$in the other next half, it is obliged to exploit in the search space. According to [33], the parameter $\vec{A}$ singly could not avoid getting stuck in the local optima. The parameter $\vec{C}$ which contains a random value in [0, 2], is introduced to solve this problem. Actually, setting a random value for the parameter $\vec{C}$ provides exploration for the GWO algorithm over the course of iteration, which is not limited only to initial iterations.

2.2.1 Improved grey wolf optimizer (IGWO)

According to [11], the exploration procedure of the GWO algorithm is nonlinear and intricate, hence the simple linear parameter $\vec{a}$ cannot provide an efficient balance between the exploration and exploitation in GWO algorithm. Therefore a nonlinear parameter is proposed based on the cosine function. The adaptive parameter $\vec{a}$ is regulated in Eq. (18).

$$\vec{a} = 1 - \cos\left(\left(1 - \frac{t}{t_{max}}\right)^k .\pi\right) \tag{18}$$

where t and t_{max} represent the current iteration and maximum number of iterations, respectively and k is a nonlinear adjustment parameter.

2.3 Simulator

The robotics simulator is used to create an application for a physical robot without depending on the real machine and reduces the cost and time. The simulator employs the visual models of the working environment and robots. The Khepera IV model is considered as the mobile robot, and Hokuyo (URG-04LX-UG01) model is considered as the laser range finder scanner sensor in the investigation.

2.3.1 Khepera IV

The Khepera IV is a differential wheeled robot whose movement is dependant on two separately driven wheels, which are located on each side of its body. This is a newer version of the Khepera family that is developed by the K-Team company. It is suitable for research and educational purposes. Khepera IV has a modular configuration with a cylindrical shape to minimize the damage under collision conditions. It offers many advanced features such as eight infra-red sensors, five ultrasonic sensors, accelerometer, gyroscope, microphone, a large duration battery, wifi and Bluetooth communications, and a color camera with 800 MHz ARM Cortex-A8 Processor.

2.3.2 Laser range finder

The Hokuyo (URG-04LX-UG01) laser range finder (LRF) scans the environment by emitting the pulse laser beams. The sensor is mounted on the top of the Khepera IV robot. If the emitted pulse contacts with an object, the reflected laser pulse is detected by a photodiode. The LRF scans the environment in a circular way with the maximum wide range of 3600 mm × 240°. Figure 1 illustrates the laser beams in a wide range in the V-REP simulator. The sensor retrieves the distance and the angle of beams, if each beam collides with the obstacles. It is notable that the angle between the two beams is 0.352° and the total number of beams is exactly 683. As can be seen from the Fig. 1, the sensor can detect the surroundings from −120° to +120°. If a robot head is aligned to x-axis of the coordination system, a full scan by sensor consists of 683 values $D_j = (d_{(1)}, d_{(2)}, \ldots d_{(683)})$ corresponding to the angle $\Phi_j = (\Phi_{(1)}, \Phi_{(2)}, \ldots \Phi_{(683)}) = (-120, -119.64, \ldots, 0, \ldots, 119.64, 120)$, where D represents the set of distances

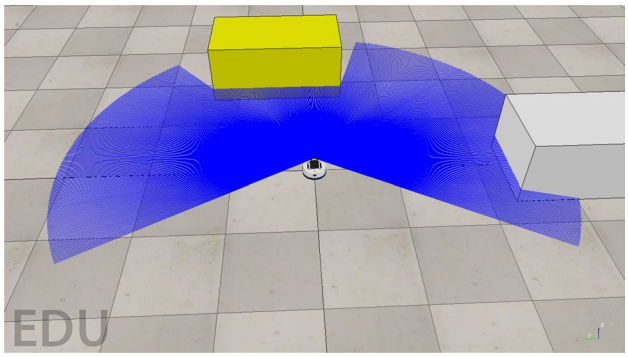


Fig. 1 Full wide range scanning by LRF

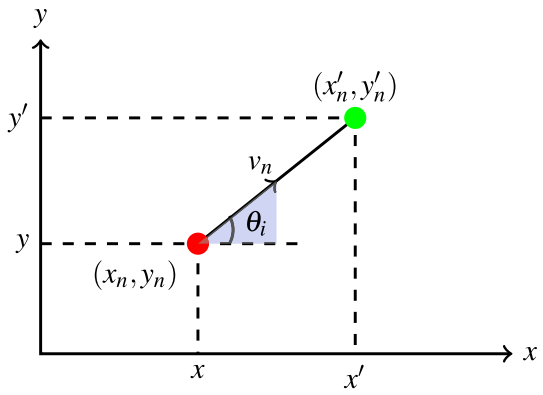


Fig. 2 Current and next position of *n*th robot

from the obstacles measured by the *j*th beam. If $d_j = 0$, it means that no collision is found by *j*th beam.

3 Problem description

3.1 Environment details

In this paper, the navigation is done in an unknown environment, there is no information about the location of obstacles before the initial motion of the robot, and the local navigation algorithms use the sensor's data directly. It is supposed that the searched area has boundaries that are completely known and each robot is aware of its start and goal positions.

3.2 Problem definition

Generally, in local path planning problems, each robot attempts to determine the next collision-free location in an unknown environment from an initial position to the final position in steps. The robot localization used in this research is based on the relative localization, where each robot calculates its next position based on the current location, angle, and velocity on given period time, which is illustrated in Fig. 2.

The next position of the robot is formulated in Eqs. (19) and (20):

$$x_n^{next} = x_n^{current} + v_n * \cos\theta_i \tag{19}$$

$$y_n^{next} = y_n^{current} + v_n * \sin\theta_i \tag{20}$$

where $(x_n^{current}, y_n^{current})$ and (x_n^{next}, y_n^{next}) are the coordination of the current position and the next position of *n*th robot in Cartesian coordinate system, respectively. Also the velocity of *n*th robot is indicated by v_n and θ_i represents the angle to direct *n*th robot.

To control the multi-robot path-planning problems, we need some criteria and rules which are explained as follows:

1. The initial and the final positions of each robot are predefined.
2. The robots use their laser range finder sensor to detect the surrounding environment before the movements.
3. The maximum robot movement in each step is dependent on the radial detection of the sensor and the fixed velocity of a robot.
4. The path navigation from the start status to the final status is done in several steps by each robot.
5. If the distance between the current position and the goal position is in the sensor detection area and there is no obstacle in the way, the robot goes to the goal position directly.

4 Methodology

In this research, the distributed multi-robot path planning problem is mainly done in three stages. All stages which are introduced here, operate for each robot individually. Firstly, the laser range finder (LRF) sensor, which is mounted on the top of robots, attempts to sense the environment based on sensing range limitation. The sensing is operated one time in each step before the robot starts to move. In the second stage, the current position of the robot and the sensor output are considered as the inputs for the procedure of the IGWO (IPSO) algorithms. Further, the multi-objective function is proposed in IGWO (IPSO) algorithms to evaluate the candidate solutions and enable them to converge to the optimal solution. The last stage directs the robot to the best position that has been founded in the second stage and updates the robot's current position. The stages are repeated until the robot reaches the goal position(see Algorithm 1).

Algorithm 1: Procedure of path planner

Input : $(x_n^{Start}, y_n^{Start}), (x_n^{goal}, y_n^{goal}), v_n$

Output: The optimal path from the current position to the next position until reaching the goal

$x_n^{current} \leftarrow x_n^{Start}, y_n^{current} \leftarrow y_n^{Start}$
while $(x_n^{current} \neq x_n^{goal}$ *and* $y_n^{current} \neq y_n^{goal})$ **do**
    LRF sensor scanning environment
    Call procedure IGWO (IPSO)
    The robot attempts to align its heading directly toward the next position by *Turn* radian.
    Go to the next position $(x_n^{current}, y_n^{current})$
    $x_n^{current} \leftarrow x_n^{next}, y_n^{current} \leftarrow y_n^{next}$
end

4.1 Proposed approach

There are three main parts of our proposed approach for improving the results of robot path planning in an unknown environment:

1. A reduction approach is applied to LRF data in order to reduce the computational time.
2. Three new objective functions are added to a custom objective function.
3. A parameter is adjusted to slightly extend the angle of the collision zone in the objective function in order to avoid the collisions.

4.2 LRF data reduction

As mentioned previously in the Sect. 2.2, after scanning the environment by LRF sensor, 683 distances are measured and their corresponding angles are stored into two separate arrays D and Φ, respectively. Since the zero value for each element in array D indicates no collision for *j*th beam, the zero value element is omitted in both arrays D and its corresponding Φ in the proposed method. According to the reduced array Φ, all the maintained elements indicate the angle of collision for the robot. In addition, to decrease the computation in the second stage, all the continuous angles are reduced to pair values, which are the first and last angles. Also, this reduction is applied in the corresponding array D. Finally, new arrays included some pair values are considered as arrays D_{obs} and Φ_{obs} for the next stage.

4.3 Procedure of IGWO (IPSO) algorithms

In this stage, the current position $(x_n^{current}, y_n^{current})$, the final position (x_n^{goal}, y_n^{goal}), and arrays D_{obs-n}, and Φ_{obs-n} are obtained from the previous stage. Moreover, the environment boundary and the robot velocity are considered as inputs for the procedure of IGWO (IPSO) algorithms (see Algorithms 2 and 3).

4.3.1 Generating initial population in IGWO (IPSO) algorithms

According to the local path planning method in this research and because of limitations on the sensing range of the sensor, the initial positions of candidate solutions are considered around the robot's current position. Hence, the candidates are distributed in a circle, where the robot's current position and the velocity are regarded as its center and radius, respectively. In the proposed method, the radius of the circle is assumed less than the LRF sensing range and is equal to the robot velocity. Consequently, the robot movement in each step is limited to the circle's interior space. The candidate position is checked to not exceed the boundary during the optimization.

4.3.2 Multiple objective fitness function

In the real path planning problems, a single objective function could not find an optimal solution based on several conditions and constraints. For instance, using one single objective function in terms of minimizing the path length in an environment with obstacles, would not satisfy other criteria such as collision-free trajectory during the optimization. So, three new objective functions are introduced in our proposed method to satisfy several criteria such as short optimal path, collision-free, and smooth path. Also, reducing the number of robots head turns is considerable in terms of energy saving.

The first objective function is employed to evaluate the candidate solutions with respect to the shortest distance criteria [30] as follows,

$$F_1 = \sqrt{\left(x_i^{cs} - x_n^{goal}\right)^2 + \left(y_i^{cs} - y_n^{goal}\right)^2} \tag{21}$$

where (x_i^{cs}, y_i^{cs}) are the positions of candidate solutions in two-dimensional coordinate system. The second objective function evaluates the candidate solution's angle and penalizes those candidate solutions which are in collision zone. This function is obtained by sensor in Eq. (22).

$$F_2 = \begin{cases} \text{PN} & \text{if } \phi_{k1} \pm \mu < \theta_i^{cs} < \phi_{k2} \pm \mu \\ 0 & \text{else} \end{cases} \tag{22}$$

where PN is a positive number to penalize the candidate solutions whose angle is in the collision zone. θ_i^{cs} represents the candidate solutions angle along the X-axis. The k pair angle elements ϕ_{k1} and ϕ_{k2} in the array $\boldsymbol{\Phi}_{obs}$ determine the collision zones obtained by the sensor with some reduction. Despite the candidate solutions whose angle is not in collision zones and are not penalized, some of them which are located near the obstacles are not considered as proper solutions. The parameter μ is adjusted in (0.1, 0.2) radians to extend the angle of collision zones in order to avoid the collision. It scans the environment with a wide range of 240 degrees according to LRF sensor, so the candidate solutions are also penalized with a positive number where their angle is not in LRF sensing wide range.

The third fitness function is used to evaluate the distance between the candidate solution and the obstacle. To find out the distance, we need to determine the collision points in Eqs. (23) and (24).

$$x_k^{obs} = x_n^{current} + d_{k1} * cos(\phi_{k1}) \tag{23}$$

$$y_k^{obs} = y_n^{current} + d_{k2} * sin(\phi_{k2}) \tag{24}$$

where (x_k^{obs}, y_k^{obs}) are the positions of collision points in two-dimensional coordinate system. d_{k1} and d_{k2} are the k pair distance elements in array D_{obs} which are obtained by the sensor. Then each candidate solution is evaluated in Eq. (25).

$$F_3 = \begin{cases} \text{PN} & \text{if } \sqrt{\left(x_i^{cs} - x_k^{obs}\right)^2 + \left(y_i^{cs} - y_k^{obs}\right)^2} < \epsilon \\ 0 & \text{else} \end{cases} \tag{25}$$

where ϵ is the minimum allowable distance from the obstacles.

The robot movement in each step is limited to the supposed circle around it. In some situations, the swarm intelligence algorithms like PSO and GWO get stuck in local optima. Consequently, the robot movement in each step becomes much less than it could traverse. Therefore, to avoid getting stuck in the local optima, the hypothetical point is assumed in the line of each candidate solution angle. The hypothetical point is calculated in Eqs. (26) and (27).

$$x_i^{hyp} = x_n^{current} + v_n * cos\left(\theta_i^{cs}\right) \tag{26}$$

$$y_i^{hyp} = y_n^{current} + v_n * sin\left(\theta_i^{cs}\right) \tag{27}$$

where (x_i^{hyp}, y_i^{hyp}) is the position of a hypothetical point in a two-dimensional coordinate system. The distance between each candidate solution and its related hypothetical point is measured in Eq. (28).

$$Dis_{c\_h} = \sqrt{\left(x_i^{cs} - x_i^{hyp}\right)^2 + \left(y_i^{cs} - y_i^{hyp}\right)^2}). \tag{28}$$

Finally, the candidate solutions which are relatively close to the robot's current position are penalized in Eq.(29).

$$F4 = 5 * \left(\frac{Dis_{c\_h}}{MaxMove - Dis_{c\_h}}\right) \tag{29}$$

where *MaxMove* indicates the maximum allowable movement for the robot in each step and it is equal to the robot velocity in this paper.

Path planning problem is considered as an optimization problem with regard to the multiple objective fitness functions. The optimization problem is stated in terms of minimization in this research. The criterion of path shortness, path smoothness, and collision avoidance path planning are obtained by the sum of four objective functions in a single conventional function by a simple additive weighting method in Eq. (30).

$$Fitness = \lambda_1 F_1 + \lambda_2 F_2 + \lambda_3 F_3 + \lambda_4 F_4 \tag{30}$$

where λ_i is the weight of *i*th objective function and it is set to 0.25.

Algorithm 2: Procedure of IGWO algorithm

Input : $(x_n^{current}, y_n^{current}), (x_n^{goal}, y_n^{goal}), D_{obs-n}, \Phi_{obs-n}, v_n, environment - boundary$

Output: $x_n^{next} \leftarrow X_\alpha^1, y_n^{next} \leftarrow X_\alpha^2, Turn \leftarrow \theta_\alpha$

Generate the initial grey wolf population $X_i (i = 1, 2...N)$ randomly around the robot sensing range
Initialize a, A and C
Evaluate the initial wolves fitness value
X_α,X_β and X_δ are the best, the second best and the third best wolves
while $I < MaxIter$ **do**
    for $i \leftarrow 1$ **to** N **do**
        update the position of ith wolf by Eq.(17) and check the wolves position(X_i^1, X_i^2) to not exceed the boundary
    end
    update a, A and C by Eq.(18),Eq.(12) and Eq.(13)
    calculate the angle between the robot's current position and wolves(candidate solutions) along the positive or negative x-axis by $\Theta_i = atan2(y, x)$ $y = X_i^2 - y_i^{current}, x = X_i^1 - x_i^{current}$
    Evaluate the initial wolves fitness value by Eq.(30)
    update X_α,X_β and X_δ
    I=I+1
end

5 Experiments and results

In this section, the environment of the multi-robot path planning problem is described and the parameters value of IGWO (IPSO)algorithms are determined. Moreover, the evaluation criteria which should be used in assessing the proposed method are expressed. Eventually, the results obtained by employing the proposed method are discussed.

5.1 Environment description

In this research, the multi-robot path planning implementation is done in V-REP simulator software. All algorithms are implemented in Lua programming language on an intel core i7-6700HQ microprocessor. The size of the environment maps is $10 \times 10\ m^2$. Each map contains a number of different static obstacles and robots.

Algorithm 3: Procedure of IPSO algorithm

Input : $(x_n^{current}, y_n^{current}), (x_n^{goal}, y_n^{goal}), D_{obs-n}, \Phi_{obs-n}, v_n, environment - boundary$

Output: $x_n^{next} \leftarrow X_{Gbest}^1, y_n^{next} \leftarrow X_{Gbest}^2 Turn \leftarrow \theta_{Gbest}$

```
Set the parameters: W_min,W_max,C_1i,C_1f,C_2i,C_2f, MaxIter,N
Generate the initial particles position randomly
Initialize the particle velocity and Pbest_i
Evaluate the initial particle fitness value
Find out Gbest
for t ← 1 to MaxIter do
    Compute C1 and C2 using Eq.(8) and Eq.(9), respectively
    for i ← 1 to N do
        Calculate Dist_i using Eq.(6)
    end
    Max_Dist ← Argmax(Dist_i)
    for i ← 1 to N do
        Calculate W_i using Eq.(5)
        for d ← 1 to 2 do
            Update V_i^d(t) according to Eq.(1)
            Update X_i^d(t) according to Eq.(2)
            Check the particles position(X_i^d(t)) to not exceed the boundary
        end
    end
    for i ← 1 to N do
        Calculate the angle between the robot's current position and particles(candidate
          solutions) along the positive or negative x-axis by Θ_i = atan2(y,x)
          y = X_i^2 − y_n^current, x = X_i^1 − x_n^current
        Evaluate each particle fitness by Eq.(30)
        Update Pbest_i and Gbest using Eq.(3) and Eq.(4), respectively
    end
end
```

5.2 Evaluation criteria

5.2.1 Total travelled path deviation (TTPD)

Suppose T_{nj} is a collision-free path which is generated by the robot R_n from a predefined initial position to a goal position G_n in jth separated execution of path planner algorithm. Consider $T_{n1}, T_{n2}, \ldots, T_{nj}$ are the travelled paths achieved by nth robot in j separated execution of the path planner algorithm. The average travelled path (ATP) of robot R_n can be represented by $(\sum_{j=1}^{k} T_{nj})/k$. Then the travelled path deviation (TPD) is calculated in Eq. (31).

$$R_{n\_TPD} = \left| R_{n\_EDist} - \frac{\sum_{j=1}^{k} T_{nj}}{k} \right| \tag{31}$$

where R_{n_EDist} is the Euclidean distance between the initial position of the robot R_n to the goal position G_n. The absolute difference between R_{n_EDist} and ATP indicates the deviation measure. Eventually, the total travelled path deviation (TTPD) for n robots is given by $\sum_{n=1}^{N} R_{n\_TPD}$.

5.2.2 Average untravelled target distance (AUTD)

Consider C_n and G_n are the current position and the goal position of the robot R_n, where C_n and G_n are the two-dimensional vectors [7]. The untravelled target distance of the robot R_n is computed in Eq. (32).

$$R_{n\_UTD} = \|C_n - G_n\| \tag{32}$$

where R_{n_UTD} is the Euclidean distance of the robot R_n between C_n and G_n. Therefore, the total untravelled target distance for n robots is given by $\sum_{n=1}^{N} R_{n\_UTD}$. The average untravelled target distance (AUTD) after k execution of the program is computed in Eq. (33).

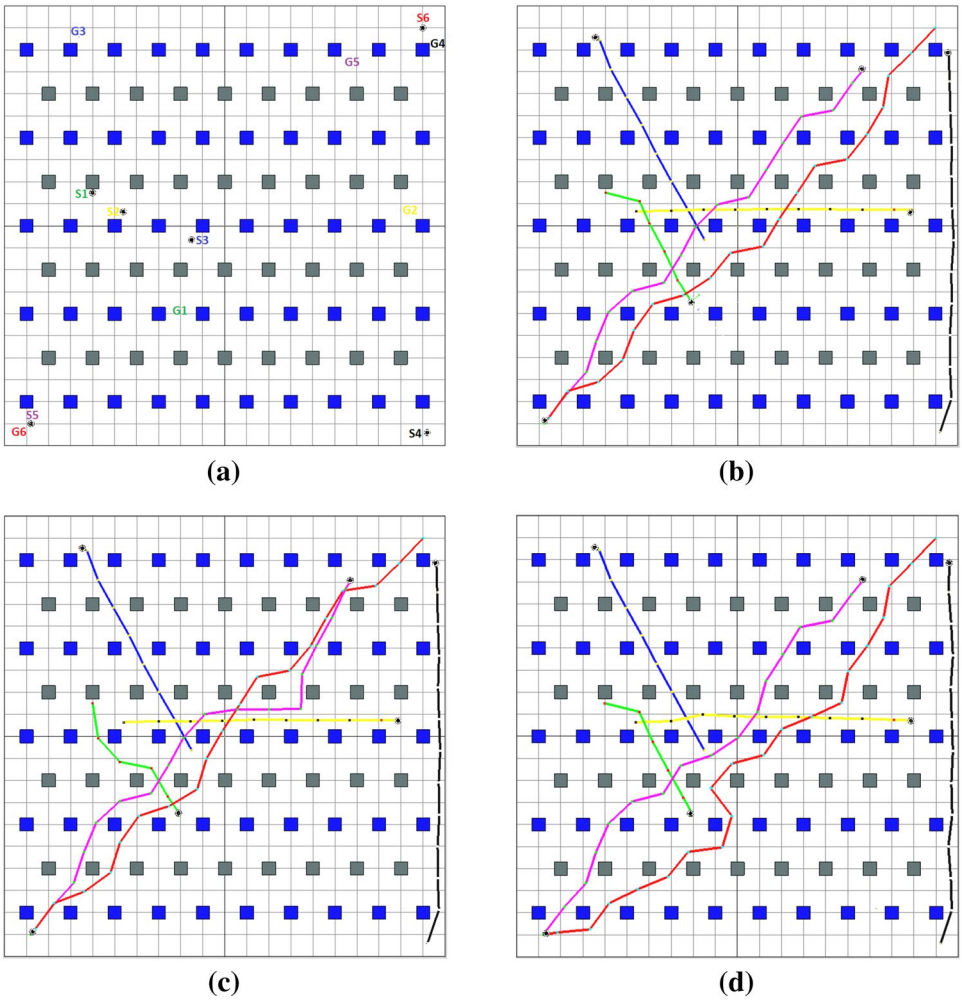


Fig. 3 Paths traversed by the robots using different algorithms in Map1. **a** Initial configuration, **b** using IGWO, **c** using GWO, **d** using IPSO

Table 1 Robots coordinate location in Map1

Robot	Initial position	Goal position	Path color
R_1	$S1(3, -0.75)$	$G1(1, 1.8)$	Green
R_2	$S2(2.3, -0.32)$	$G2(-4, -0.36)$	Yellow
R_3	$S3(0.75, 0.32)$	$G3(3.3, -4.3)$	Blue
R_4	$S4(-4.6, 4.7)$	$G4(-4.8, -4)$	Black
R_5	$S5(4.4, 4.5)$	$G5(-2.9, -3.6)$	Purple
R_6	$S6(-4.5, -4.5)$	$G6(4.4, 4.5)$	Red

$$AUTD = \frac{\sum_{n=1}^{N} R_{n\_UTD}}{k} \tag{33}$$

5.3 Comparison between different algorithms

We design Map1 to test our proposed method by using different algorithms such as IGWO, GWO, and IPSO. The initial configuration of Map1 in the V-REP simulator is shown in Fig. 3a, which includes 6 robots and 86 obstacles. The obstacles are static and their size is similar. The coordinate location of the robots and their goals are represented in Table 1.

The velocity of the robots is not altered during the execution and it is set to 0.8 m/s. The paths traversed by the robots utilizing IGWO, GWO, and IPSO algorithms are illustrated in Fig. 3b, c, d. The LRF sensing wide range is assigned to 1 m × 240°. The value of parameters PN and ϵ in the objective functions is equal to 10 and 0.25 m, respectively. The number of candidate solutions and iterations are adjusted to N = 50 and MaxIter = 40 in all three algorithms. It is worth mentioning that the best value of the population size has been found empirically and based on problem dimension. It is fixed to 50 in order to balance the computational complexity and convergence rate. The value of parameters W_{min} and W_{max} is assigned to 0.4 and 0.9 in the IPSO algorithm. Eventually, the value of C_{1i}, C_{1f}, C_{2i} and C_{2f} is set to 2.5, 0.5, 0.5 and 2.5, respectively. It should be noted that the value of each parameter varies over a given range while the rest of them are fixed at their

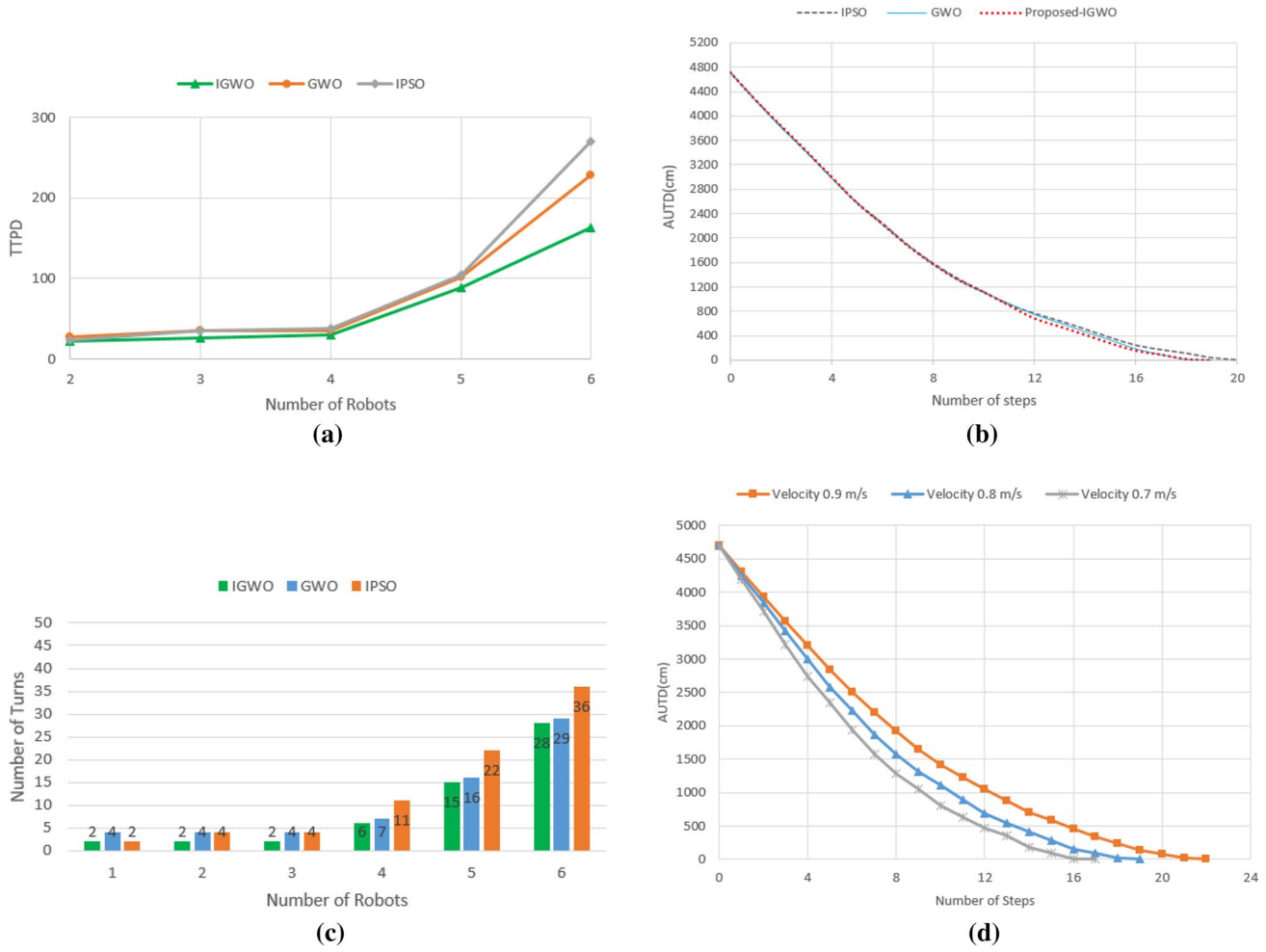


Fig. 4 Path planning using IGWO, GWO and IPSO in Map1. **a** TTPD versus the number of robots. **b** AUTD versus the number of steps have taken. **c** Total number of turns versus the number of robots. **d** AUTD versus the number of steps have taken by varying the velocities of robots

Table 2 Comparison between IGWO, GWO and IPSO in Map1

No. of robots	No. of obstacles	No. of steps			Length of path (m)			TTPD (cm)			Simulation time (min)		
		IGWO	GWO	IPSO	IGWO	GWO	IPSO	IGWO	GWO	IPSO	IGWO	GWO	IPSO
2	28	14	14	14	**9.77**	9.81	9.80	**22**	27	23	2.27	2.24	**1.38**
3	28	21	21	21	**15.16**	15.17	15.17	**26**	35	35	2.27	2.24	**1.38**
4	28	33	33	33	**23.83**	23.87	23.89	**31**	35	37	3.10	3.08	**2.36**
5	28	49	49	49	**35.34**	35.44	35.46	**89**	101	104	5.12	5.01	**4.12**
6	28	**67**	**67**	69	**48.71**	49.37	49.78	**164**	229	270	5.58	5.45	**5.20**

final values. It means that we search the optimal parameters empirically and report the best results.

According to Fig. 4a, it can be noticed that by increasing the number of robots, the value of total travelled path deviation for IGWO is less compared with GWO and IPSO algorithms. It can be noted from Fig. 4a that the performance of the IGWO algorithm in determining the next position is more reliable than GWO and IPSO. On the other hand, the comparison of AUTD versus the number of steps is illustrated in Fig. 4b. By increasing the steps, AUTD converges to the zero value. The behavior of all three

Table 3 Parameters and values assigned for IGWO and path planner algorithms in Map2 and Map3

Parameters	Map2	Map3
Wolf population	50	50
Maximum iteration	40	40
LRF wide range	1.5 m	0.8 m
Robot velocity	0.8 m/s	0.8 m/s
PN	10	10
ϵ	0.25 m	0.23 m

algorithms is similar until the step 11, but IGWO shows a better performance in the last steps compared to GWO and IPSO.

One of the most important evaluation criteria in the robot path planning problems is energy consumption. Most the robot's energy consumption is due to kinetic energy. The total kinetic energy is simply the sum of translational and rotational energy. In two-wheeled mobile robot, the rotational motion consumes more energy than translational motion. Therefore optimizing the rotational motion is really important in terms of saving energy. The number of turns is increased when the angle of the robot's next position is not in line with the angle of the current position. The number of turns required for IGWO, GWO, and IPSO is illustrated in Fig. 4c. It is obvious that the number of turns by IGWO algorithm is fewer than it in the other algorithms. It can be observed from Fig. 4d that by decreasing the velocity of the robots, AUTD requires more steps to reach the zero value.

The other results obtained by the execution of multi-robot path planning problem in Map1 are presented in Table 2. The parameters such as path length, number of steps, TTPD, and execution time have been discussed for the comparison between various algorithms. The best results are boldfaced in the Table 2. In terms of the number of steps taken by the robots, all three algorithms have the same performance. By adding the latest robot, IGWO and GWO need the fewer steps than IPSO. The length of the actual path traveled by the robots is another factor in comparing the algorithms. From Table 2, it can be seen that the IGWO algorithm achieves the better results in the length of the path. Similarly, in terms of TTPD, the paths traversed by the robots with IGWO algorithm have less deviation compared to GWO and IPSO algorithms.

In the V-REP simulator, the calculation time of the various sensors, dynamic modules, and the rendering time increase the overall simulation time. Therefore, the path planing problem with more obstacles and robots needs more time to be executed. From Table 2, it can be seen that IPSO is less time consuming than IGWO and GWO algorithms.

Table 4 Robots coordinate location in Map2

Robot	Initial position	Goal position	Path color
R_1	$S1(0, 0)$	$G1(-3, 3)$	Green
R_2	$S2(2.5, -2)$	$G2(3, 0.2)$	Yellow
R_3	$S3(-3, -3)$	$G3(-1, 2)$	Dark blue
R_4	$S4(0, -2)$	$G4(3.5, 3.5)$	Black
R_5	$S5(-4, 4.5)$	$G5(-3.2, -2.7)$	Purple
R_6	$S6(-0.9, -4.3)$	$G6(-3.5, 1.5)$	Red
R_7	$S7(-4, -4.8)$	$G7(4, -2)$	Light blue
R_8	$S8(4.5, 4.5)$	$G8(-2.5, -3.5)$	Orange

5.4 Computational complexity and execution time analysis

The proposed IGWO described in Algorithm 2 consumes times in some different parts. First of all, to initialize the population of N candidate solutions and the parameters of IGWO (a, A and C), O(N) and O(1) operations are needed, respectively. After that, calculating the first three best solutions (X_α, X_β and X_δ) would take O(N). The next steps are conducted inside the while loop. The for loop requires O(N) operations to update the position of population. Also, updating the parameters of IGWO (a, A and C) and calculating the angle between the robot's current position and the wolves would take O(1). In the next step, calculating the fitness of each candidate solution needs O(N) operations. Finally, we would need O(N) operations to update X_α, X_β and X_δ. $O(MaxIter \times (N + 1 + 1 + N + N))$ operations are required in the while loop, where MaxIter displays the maximum number of iterations. Hence, the overall computational complexity of the algorithm is the summation of the above time complexities which is $O(3 \times MaxIter \times N + 2 \times N + MaxIter + 1)$. In summary, the number of operations can be further simplified to $O(MaxIter \times N)$.

Computing the execution time of IGWO is also done in various parts. Firstly, initializing the parameters requires $O(2 \times N)$ time, where N represents the population size and the dimension of the problem is equal to 2. Then, the calculation of the control parameters of the IGWO needs

Table 5 Robots coordinate location in Map3

Robot	Initial position	Goal position	Path color
R_1	$S1(4, -4.5)$	$G1(1, 0.5)$	Green
R_2	$S2(-1.3, -2)$	$G2(1.8, 1.3)$	Yellow
R_3	$S3(1.7, -4.5)$	$G3(-2.9, -4.5)$	Dark blue
R_4	$S4(4.4, 4.5)$	$G4(-4.17, -1.3)$	Black
R_5	$S5(-4.9, -4.9)$	$G5(4.4, 4.5)$	Purple

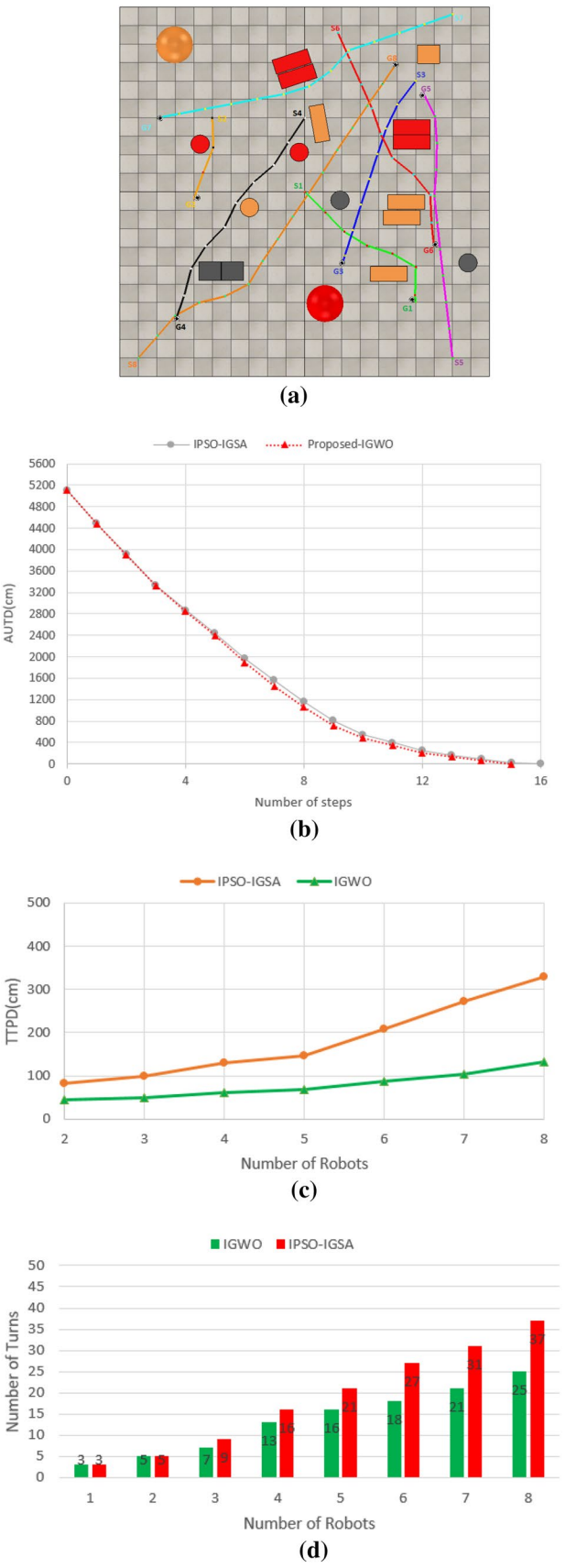


Fig. 5 Path planning using IGWO and IPSO-IGSA in Map2. **a** Paths traversed by the robots using IGWO algorithm. **b** AUTD versus the number of steps have taken by different velocities of robots. **c** TTPD versus the number of robots. **d** Total number of turns versus the number of robots

$O(2 \times N)$ time. To update the robots position-updated equations in IGWO, $O(2 \times N)$ time would require. Also, evaluation of the fitness value of each robot takes $O(2 \times N)$ time. In summary, the total time execution of the IGWO is $O(2 \times N \times MaxIter)$, where MaxIter idicates the maximum number of iterations.

5.5 Comparison with another research

Finally, our proposed method using IGWO is compared to improved particle swarm optimization-improved gravitational search algorithm (IPSO-IGSA) [7] in the same conditions and environment to evaluate its performance. Two maps with different obstacles are considered for this comparison. Figures 5a and 6a show Map2 and Map3, where the color paths are traversed by the robots using IGWO algorithm. The parameters considered for IGWO and path planner algorithms are represented in Table 3. The initial and goal positions of the robots in Map1 and Map2 are presented in Tables 4 and 5, respectively.

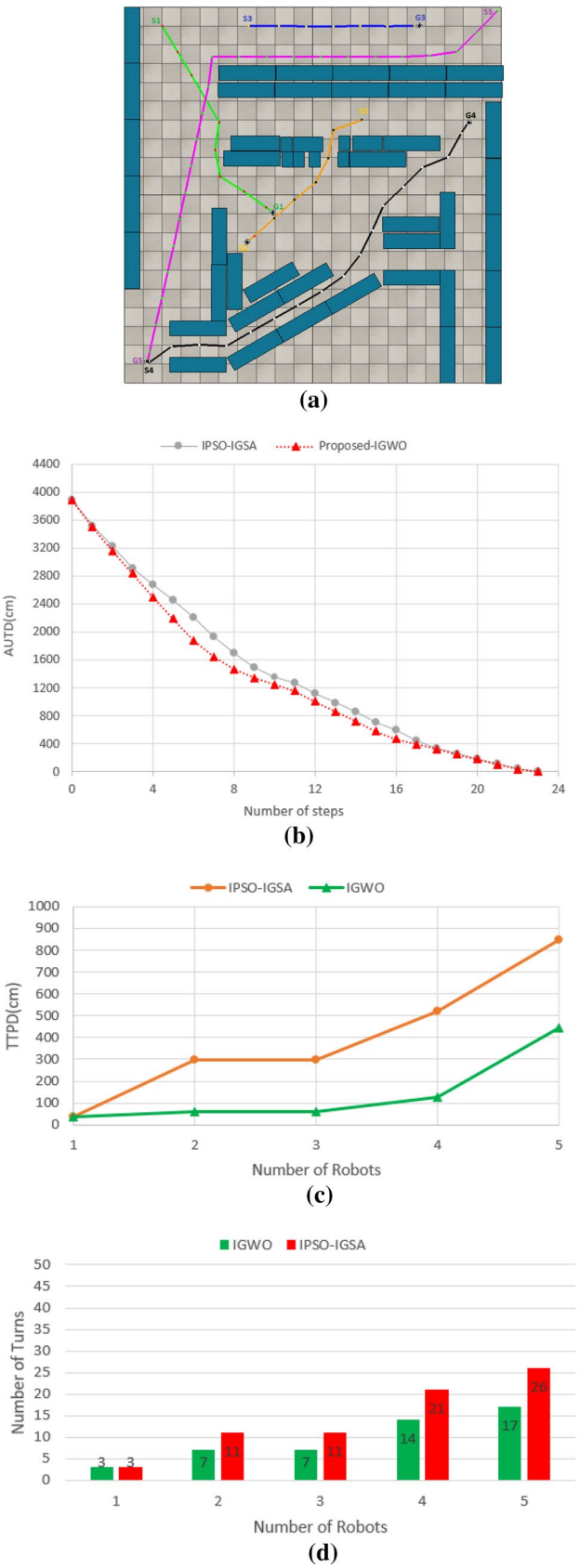
Figures 5b and 6b show that the value of AUTD by IGWO algorithm in each step is lower than it in IPSO-IGSA. Similarly, the deviation in the proposed method is less than it in IPSO-IGSA algorithm based on Figs. 5c and 6c. Furthermore, the number of robots head turns in Figs. 5d and 6d is less than another algorithm using the IGWO algorithm.

Finally, the results of the experiments are presented in Tables 6 and 7 in terms of four performance factors, including the number of steps to reach the goal, length of the path, TTPD and simulation time. The best results are boldfaced in the Tables 6 and 7. The comparison results demonstrate that the proposed IGWO performs better than IPSO-IGSA in both Map2 and Map3.

5.6 Analysis of the proposed method

The results shown in the previous section indicate that the proposed methods in the structure of IGWO could obtain a short, safe and smooth path for a robot in an unknown environment. The objective functions play a major role in finding a suitable path. The F_1 objective function causes IGWO to find the solution that has a minimum distance to the goal by providing a good balance between the exploration and exploitation. The length of the path in Tables 2, 6 and 7 shows the importance of this function. However, this function alone cannot guarantee to reach

Fig. 6 Path planning using IGWO and IPSO-IGSA in Map3. **a** Paths traversed by the robots using IGWO algorithm. **b** AUTD versus the number of steps have taken by different velocities of robots. **c** TTPD versus the number of robots. **d** Total number of turns versus the number of robots



a collision-free path because there are some obstacles in the environment. Therefore, F_2 function is proposed to encourage IGWO to discover the solutions that are in free-collision zone by the sensor and avoid the collision. Also, it contributes the robot to turn its head fewer according to Figs. 4c, 5d and 6d. In addition, the deviation is reduced by fewer turnings and the path is traversed smoother. However, F_1 and F_2 objective functions are not sufficient for the real path planning problem in an unknown environment because IGWO may converge to a solution that is exactly near the obstacle. Hence, F_3 is introduced to penalize the solutions that their distances are lower than a defined threshold to avoid the conflict with the obstacles. In some situations in which the path is traversed by the robot R2 and R4 in the Map 6a, the obstacles are close to each other and IGWO may get stuck in the solutions that are near the robot. Consequently, the last objective function F_4 causes IGWO to converge to the solutions that are in the farthest allowable distance in each step. By introducing F_4 objective function, the number of steps in Tables 2, 6 and 7 is reduced because the path planner algorithm calls the IGWO algorithm fewer.

The main purpose for using GWO in the path planning problem is its considerable advantages. One of the important advantages of GWO is its easy adaptability to different optimization problems. Also, the acquired information is not required for the initial search and flexibility and a few parameters are needed to be initialized in GWO. However, GWO has some disadvantages such as slow and premature convergence, and deficient local search ability. Due to the drawbacks of original GWO, an improved version of it (IGWO) is used in this paper. The nonlinear parameter a is used in IGWO which is changed over the course of iteration in order to provide a better global and local exploration of GWO algorithm. Based on the results obtained in the previous section, the IGWO has a better performance compared to GWO, IPSO, and IPSO-IGSA algorithms.

6 Conclusion and future works

In this research, the path planning problem is done autonomously by the robot due to the metaheuristic characteristics of the swarm intelligence algorithms to find a global optimal solution. The robots navigate the environment in the local planning approach by using received data from

Table 6 Comparison between IGWO and IPSO-IGSA in Map2

No. of robots	No. of obstacles	No. of steps		Length of path (m)		TTPD (cm)		Simulation time (min)	
		IGWO	IPSO-IGSA	IGWO	IPSO-IGSA	IGWO	IPSO-IGSA	IGWO	IPSO-IGSA
2	18	10	10	**6.94**	7.32	**45**	83	**0.57**	1.40
3	18	18	18	**12.36**	12.85	**49**	98	**1.07**	2.05
4	18	27	27	**18.99**	19.67	**61**	129	**1.33**	2.42
5	18	37	37	**26.29**	27.08	**67**	146	**1.51**	3.01
6	18	**46**	47	**32.84**	34.05	**87**	208	**2.10**	3.20
7	18	**58**	60	**41.48**	43.16	**103**	271	**2.29**	3.40
8	18	**73**	76	**52.39**	54.36	**132**	329	**3.23**	4.47

Table 7 Comparison between IGWO and IPSO-IGSA in Map3

No. of robots	No. of obstacles	No. of steps		Length of path (m)		TTPD (cm)		Simulation time (min)	
		IGWO	IPSO-IGSA	IGWO	IPSO-IGSA	IGWO	IPSO-IGSA	IGWO	IPSO-IGSA
1	49	9	9	**6.20**	6.21	**37**	38	**1.28**	2.30
2	49	**16**	19	**10.90**	13.32	**62**	298	**1.58**	2.52
3	49	**23**	26	**15.50**	17.92	**62**	298	**2.10**	3.14
4	49	**40**	44	**26.88**	30.88	**125**	519	**4**	5.12
5	49	**63**	67	**43.27**	47.37	**443**	846	**5.29**	6.40

the sensors. The distributed approach is utilized for multi-robot path planning problem in our proposed methods. The main purpose of this research is discovering a free-collision, smooth, and optimized short path as well as reducing the number of robots head turns to save energy and time. To satisfy this criterion, we proposed three objective functions with an avoidance collision approach. First, the proposed method was applied in GWO, IGWO and IPSO, then we made a comparison between the algorithms to find a shorter, safer and smoother path. The results show the superiority of IGWO compared to other algorithms due to better local and global exploration, and local minima avoidance. Finally, the proposed methods applied in IGWO are compared to IPSO-IGSA. The outcomes illustrate that the proposed methods in IGWO have a better performance in comparison with IPSO-IGSA. Therefore, the proposed methods alongside IGWO have more ability to find a shorter, safer and smoother path. Moreover, the execution time and the number of the robots head turns are reduced significantly in the suggested methods.

However, the proposed objective functions are dependent on the data acquired by LRF sensor and cannot work with other sensors such as the proximity sensor and so forth. Although IGWO performance is superior compared to other algorithms, it needs more enhancement to avoid the slow and premature convergence. Moreover, the coordination and cooperation between the robots are not considered and also the obstacles are static in this research. As a future work, a combination of dynamic obstacles and multi-robot path planning is our purpose. In addition, we will employ other recent algorithms such as equilibrium optimizer, whale optimization algorithm, momentum balance algorithm, emperor penguins colony, mass and energy balances algorithm, and lion optimizer to solve the path planning problem.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Asano T, Asano T, Guibas L, Hershberger J, Imai H (1985) Visibility-polygon search and euclidean shortest paths. In: 26th Annual symposium on foundations of computer science. IEEE, pp 155–164
2. Atyabi A, Phon-Amnuaisuk S, Ho CK (2010) Navigating a robotic swarm in an uncharted 2D landscape. Appl Soft Comput 10(1):149–169
3. Barraquand J, Latombe JC (1991) Robot motion planning: a distributed representation approach. Int J Robot Res 10(6):628–649
4. Bhattacharya P, Gavrilova ML (2007) Voronoi diagram in optimal path planning. In: null. IEEE, pp 38–47

5. Chakraborty J, Konar A, Jain LC, Chakraborty UK (2009) Cooperative multi-robot path planning using differential evolution. J Intell Fuzzy Syst 20(1,2):13–27
6. Chhillar A, Choudhary A (2020) Mobile robot path planning based upon updated whale optimization algorithm. In: 2020 10th International conference on cloud computing, data science and engineering (confluence). IEEE, pp 684–691
7. Das P, Behera HS, Panigrahi BK (2016) A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm Evol Comput 28:14–28
8. Dewangan RK, Shukla A, Godfrey WW (2020) A solution for priority-based multi-robot path planning problem with obstacles using ant lion optimization. Mod Phys Lett B 34:2050137
9. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, 1995. MHS'95. IEEE, pp 39–43
10. Frontzek T, Goerke N, Eckmiller R (1998) Flexible path planning for real-time applications using A*-method and neural RBF-networks. In: 1998 IEEE International conference on robotics and automation. Proceedings, vol 2. IEEE, pp 1417–1422
11. Gai W, Qu C, Liu J, Zhang J (2018) An improved grey wolf algorithm for global optimization. In: 2018 Chinese control and decision conference (CCDC). IEEE, pp 2494–2498
12. Guo Y, Parker LE (2002) A distributed and optimal motion planning approach for multiple mobile robots. In: IEEE International conference on robotics and automation, 2002. Proceedings. ICRA'02, vol 3. IEEE, pp 2612–2619
13. Gupta S, Deep K (2018) Cauchy grey wolf optimiser for continuous optimisation problems. J Exp Theor Artif Intell 30(6):1051–1075
14. Gupta S, Deep K (2019) Enhanced leadership-inspired grey wolf optimizer for global optimization problems. Eng Comput 36:1–24
15. Gupta S, Deep K (2019) Hybrid grey wolf optimizer with mutation operator. In: Bansal J, Das K, Nagar A, Deep K, Ojha A (eds) Soft computing for problem solving. Advances in intelligent systems and computing, vol 817. Springer, Singapore
16. Gupta S, Deep K (2019) A novel random walk grey wolf optimizer. Swarm Evol Comput 44:101–112
17. Gupta S, Deep K (2019) An opposition-based chaotic grey wolf optimizer for global optimisation tasks. J Exp Theor Artif Intell 31(5):751–779
18. Gupta S, Deep K (2020) Optimal coordination of overcurrent relays using improved leadership-based grey wolf optimizer. Arab J Sci Eng 45(3):2081–2091
19. Gupta S, Deep K, Assad A (2020) Reliability–redundancy allocation using random walk gray wolf optimizer. In: Das K, Bansal J, Deep K, Nagar A, Pathipooranam P, Naidu R (eds) Soft computing for problem solving. Advances in intelligent systems and computing, vol 1048. Springer, Singapore
20. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
21. Hocaoglu C, Sanderson AC (1996) Planning multi-paths using speciation in genetic algorithms. In: Proceedings of IEEE international conference on evolutionary computation, 1996. IEEE, pp 378–383
22. Hossain MA, Ferdous I (2015) Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. Robot Auton Syst 64:137–141
23. Ju MY, Wang SE, Guo JH (2014) Path planning using a hybrid evolutionary algorithm based on tree structure encoding. Sci World J 2014:746260
24. Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Robot Res 30(7):846–894
25. Kober J, Bagnell JA, Peters J (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32(11):1238–1274
26. Konar A, Goswami I, Singh SJ, Jain LC, Nagar AK (2013) A deterministic improved q-learning for path planning of a mobile robot. IEEE Trans Syst Man Cybern Syst 43(5):1141–1153
27. Li B, Liu H, Xiao D, Yu G, Zhang Y (2017) Centralized and optimal motion planning for large-scale AGV systems: a generic approach. Adv Eng Softw 106:33–46
28. Lu L, Gong D (2008) Robot path planning in unknown environments using particle swarm optimization. In: Fourth international conference on natural computation, 2008. ICNC'08, vol 4. IEEE, pp 422–426
29. Masehian E, Amin-Naseri MR (2008) Sensor-based robot motion planning—a tabu search approach. IEEE Robot Autom Mag 15(2):48–57
30. Masehian E, Sedighizadeh D (2010) A multi-objective PSO-based algorithm for robot path planning. In: 2010 IEEE International conference on industrial technology (ICIT). IEEE, pp 465–470
31. Masehian E, Sedighizadeh D (2013) An improved particle swarm optimization method for motion planning of multiple robots. In: Martinoli A et al (eds) Distributed autonomous robotic systems. Springer tracts in advanced robotics, vol 83. Springer, Berlin, Heidelberg
32. Miao H, Tian YC (2008) Robot path planning in dynamic environments using a simulated annealing based approach. In: 2008 10th International conference on control, automation, robotics and vision. IEEE, pp 1253–1258
33. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
34. Nasrollahy AZ, Javadi HHS (2009) Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target. In: 2009 Third UKSim European symposium on computer modeling and simulation. IEEE, pp 60–65
35. Noguchi N, Terao H (1997) Path planning of an agricultural mobile robot by neural network and genetic algorithm. Comput Electron Agric 18(2–3):187–204
36. Paden B, Čáp M, Yong SZ, Yershov D, Frazzoli E (2016) A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans Intell Veh 1(1):33–55
37. Parker JK, Khoogar AR, Goldberg DE (1989) Inverse kinematics of redundant robots using genetic algorithms. In: 1989 IEEE International conference on robotics and automation, 1989. Proceedings. IEEE, pp 271–276
38. Qiaorong Z, Guochang G (2008) Path planning based on improved binary particle swarm optimization algorithm. In: 2008 IEEE Conference on robotics, automation and mechatronics. IEEE, pp 462–466
39. Qin YQ, Sun DB, Li N, Cen YG (2004) Path planning for mobile robot using the particle swarm optimization with mutation operator. In: Proceedings of 2004 international conference on machine learning and cybernetics, 2004, vol 4. IEEE, pp 2473–2478
40. Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255
41. Renfrew D, Yu XH (2012) Traffic signal optimization using ant colony algorithm. In: The 2012 international joint conference on neural networks (IJCNN). IEEE, pp 1–7
42. Sadati N, Taheri J (2002) Solving robot motion planning problem using Hopfield neural network in a fuzzified environment. In: Proceedings of the 2002 IEEE international conference on fuzzy systems, 2002. FUZZ-IEEE'02, vol 2. IEEE, pp 1144–1149

43. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
44. Schouwenaars T, De Moor B, Feron E, How J (2001) Mixed integer programming for multi-vehicle path planning. In: 2001 European control conference (ECC). IEEE, pp 2603–2608
45. Šeda M (2007) Roadmap methods versus cell decomposition in robot motion planning. In: Proceedings of the 6th WSEAS international conference on signal processing, robotics and automation. World Scientific and Engineering Academy and Society (WSEAS), pp 127–132
46. Sundar S, Shiller Z (1997) Optimal obstacle avoidance based on the Hamilton–Jacobi–Bellman equation. IEEE Trans Robot Autom 13(2):305–310
47. Thida M, Eng HL, Monekosso DN, Remagnino P (2013) A particle swarm optimisation algorithm with interactive swarms for tracking multiple targets. Appl Soft Comput 13(6):3106–3117
48. van Den Berg J, Snoeyink J, Lin MC, Manocha D (2009) Centralized path planning for multiple robots: optimal decoupling into sequential plans. In: Robotics: science and systems. https://doi.org/10.15607/RSS.2009.V.018
49. Wang L, Liu Y, Deng H, Xu Y (2006) Obstacle-avoidance path planning for soccer robots using particle swarm optimization. In: IEEE International conference on robotics and biomimetics, 2006. ROBIO'06. IEEE, pp 1233–1238
50. Wu J, Wang H, Li N, Yao P, Huang Y, Su Z, Yu Y (2017) Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by adaptive grasshopper optimization algorithm. Aerosp Sci Technol 70:497–510
51. Wu KH, Chen CH, Der Lee J (1996) A cache-genetic-based modular fuzzy neural network for robot path planning. In: IEEE International conference on systems, man, and cybernetics, 1996, vol 4. IEEE, pp 3089–3094
52. Zhang MX, Yan HF, Wu JY, Zheng YJ (2020) Quarantine vehicle scheduling for transferring high-risk individuals in epidemic areas. Int J Environ Res Public Health 17(7):2275
53. Zhang S, Zhou Y, Li Z, Pan W (2016) Grey wolf optimizer for unmanned combat aerial vehicle path planning. Adv Eng Softw 99:121–136
54. Zheng YJ (2015) Water wave optimization: a new nature-inspired metaheuristic. Comput Oper Res 55:1–11
55. Zhu Z, Wang F, He S, Sun Y (2015) Global path planning of mobile robots using a memetic algorithm. Int J Syst Sci 46(11):1982–1993