Neural Computing and Applications (2021) 33:15569–15599
https://doi.org/10.1007/s00521-021-06179-0

ORIGINAL ARTICLE

Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms

Farzad Kiani[1] · Amir Seyyedabbasi[2] · Royal Aliyev[3] · Murat Ugur Gulle[3] · Hasan Basyildiz[3] · M. Ahmed Shah[4]

Received: 12 December 2020 / Accepted: 27 May 2021 / Published online: 10 June 2021

Abstract

Three-dimensional path planning for autonomous robots is a prevalent problem in mobile robotics. This paper presents three novel versions of a hybrid method designed to assist in planning such paths for these robots. In this paper, an improvement on Rapidly exploring Random Tree (RRT) algorithm, namely Adapted-RRT, is presented that uses three well-known metaheuristic algorithms, namely Grey Wolf Optimization (GWO), Incremental Grey Wolf Optimization (I-GWO), and Expanded Grey Wolf Optimization (Ex-GWO)). RRT variants, using these algorithms, are named Adapted-RRT_{GWO}, Adapted-$RRT_{I\text{-}GWO}$, and Adapted-$RRT_{Ex\text{-}GWO}$. The most significant shortcoming of the methods in the original sampling-based algorithm is their inability in finding the optimal paths. On the other hand, the metaheuristic-based algorithms are disadvantaged as they demand a predetermined knowledge of intermediate stations. This study is novel in that it uses the advantages of sampling and metaheuristic methods while eliminating their shortcomings. In these methods, two important operations (length and direction of each movement) are defined that play an important role in selecting the next stations and generating an optimal path. They try to find solutions close to the optima without collision, while providing comparatively efficient execution time and space complexities. The proposed methods have been simulated employing four different maps for three unmanned aerial vehicles, with diverse sets of starting and ending points. The results have been compared among a total of 11 algorithms. The comparison of results shows that the proposed path planning methods generally outperform various algorithms, namely BPIB-RRT*, tGSRT, GWO, I-GWO, Ex-GWO, PSO, Improved BA, and WOA. The simulation results are analysed in terms of optimal path costs, execution time, and convergence rate.

Keywords Optimal path planning · Unmanned aerial vehicle · Autonomous mobile robot · Metaheuristic algorithm

1 Introduction

In an environment comprising of numerous autonomous robots and obstacles, moving robots safely and in a collision-free manner is a pivotal task. Path planning is a problem of finding a continuous path that the robot will use to drive from the source to the destination, also known as a configuration. The purpose of motion planning in robotics is to find several valid configurations that can be employed in moving the robot from the source to the target. Unlike motion planning, one of the design goals in path planning is discovering the optimum path in terms of the least time used, along with modelling the entire environment. Path planning can be used in fully or partially known environments, as well as entirely unknown environments where

✉ Farzad Kiani
farzadkiani@arel.edu.tr

[1] Software Engineering Department, Faculty of Engineering, Istinye University, Istanbul, Turkey

[2] Computer Engineering Department, Faculty of Engineering and Architecture, Beykent University, Istanbul, Turkey

[3] Computer Engineering Department, Faculty of Engineering, Istanbul Aydin University, Istanbul, Turkey

[4] Computer Engineering Department, Faculty of Engineering, Istanbul Ayvansaray University, Istanbul, Turkey

sensed information defines the desired robot motion. Mobile robots are a cutting-edge technology that can be employed in various novel research areas such as Internet of Things (IoT) [1], military [2], and health [3]. These robots have also been used in Vehicle Ad hoc Network (VANET) [4] and Flying Ad hoc Network (FANET) [5], a subset of the Mobile Ad hoc networks, and Internet of Drone (IoD) [6] and Internet of Vehicle (IoV) [7] in the IoT category. Considering autonomous routing techniques of these systems, one of the aims is to find safe paths in the shortest possible time, using the resources effectively. These robots consist of sensors and actuators. Furthermore, each robot (node) has a processor and a memory. As such it is adequate to state that these devices can function as an all-in smart agent. Therefore, artificial intelligence-based techniques can be easily implemented using these smart nodes. When appropriated in interconnected and successive systems, it is vital to plan a path for each mobile autonomous device such as unmanned aerial vehicles (UAVs) and drones.

In unstructured environments, filled with uncertain elements, issues on the periphery of robots become increasingly difficult to resolve, causing an ever-growing demand for 3D path planning algorithms. Obviously, a simple 2D algorithm cannot qualify for planning complex situations. The difficulties of path planning in a 3D environment, unlike 2D path planning, increase exponentially due to the inherent kinematic nature of the environment. Furthermore, finding optimal 3D path planning is a Non-Deterministic Polynomial-Time (NP-hard) problem. In general, the metaheuristic methods are widely used to solve various optimization problems [8, 9]. Optimal path planning includes the shortest path length where the selected path should be as far as possible from obstacles; it must be smooth, i.e. without sharp turn, and must consider motion constraints. As such, it is pragmatic to focus on three-dimensional areas with many obstacles and find suitable routes for mobile robots. Therefore, in this paper, three-dimensional path planning methods have been made available. Hence, metaheuristic algorithms are suitable candidates for finding a solution. No-Free-Lunch (NFL) [10] asserts that there is no specific algorithm providing the best solution for every optimization problem. As such, there is a considerable demand to develop new metaheuristic algorithms that can be used in various problems. Solution techniques in the 3D path planning algorithms for autonomous robots include visibility graphs [11], probabilistic road maps [12], randomly exploring algorithms [13], random-exploring algorithms [14], and heuristics. However, literature shows that metaheuristic methods are much better at 3D path planning [15–17]. On the other hand, one of the effective methods of solving the 3D path planning problem is sampling-based methods [18]. Rapidly exploring Random Tree (RRT) [19] is one of the most famous sampling-based method. The main aim of RRT is to generate a path between source and destination stations avoiding obstacles. However, the biggest shortcoming of this algorithm is the fact that it may not find the optimum path. On the other hand, metaheuristic algorithms can be an appropriate solution to solve the optimal pathfinding.

Generally, the methods in the sampling-based category are very fast and useful in real applications; indeed, they (1) are able to find a feasible motion plan in a relatively short time and guaranteeing probabilistic completeness (the path found may not be optimal). (2) can be applied in different real areas and various dynamic models. However, there are some shortcomings. They suffer from computational overload and time inefficiency in optimal pathfinding. The other disadvantage is that it is not considered successful in finding an optimal path. In addition, they are not as successful at convergence rate, at least like metaheuristic algorithms. Another issue is that they are not very successful in complex environments [15, 18, 22].

On the other hand, in the metaheuristic-based methods, in the process of movement of each robot from the current station to the next station, the options are generally predetermined in advance and the positions of these options (candidate next stations) are also fixed. Therefore, many metaheuristic-based methods used in solving these types of problems do not deal with the angle and direction of motion and only care about the distance to travel or possible other parameters. Another important problem is that the locations and numbers of intermediate stations between the starting and destination points of each UAV are kept in memory. This may cause inefficient memory usage [18, 19].

To get around this defect, in this paper, a novel hybrid method is proposed using the advantages of sampling-based and metaheuristic-based algorithms for solving the 3D path planning problem. Three metaheuristic algorithms (Grey Wolf Optimization (GWO) [20], Incremental Grey Wolf Optimization (I-GWO) [21], and Expanded Grey Wolf Optimization (Ex-GWO) [21]) are used in the working mechanism of this method. That's why the method is presented with three distinct variants. Benefiting from them, the optimal path planning mechanisms are realized for each autonomous mobile robot in different environments, containing various obstacles without any collisions. Each proposed method attempts to find an almost optimal path by eliminating the process of creating complex environment models based on stochastic approaches. They can be faster and more successful in finding the most suitable solutions. The contribution and objective of the proposed method can be summarized as follows:

(1) It is usable in dynamic environments and real applications as fast.
(2) It tries to find collision-free optimal paths in a short time with the least process costs.
(3) It solves the problem of the excessive computational overhead of sampling-based methods in finding optimum paths. Also, unlike sampling-based methods, it is successful in finding the optimal path with a good convergence rate.
(4) It provides efficient memory usage of UAVs. Because, with this method, UAVs do not need to store predetermined static map information.
(5) Unlike metaheuristic methods, selectable intermediate nodes between target and source are not predetermined and their numbers are not fixed. Therefore, the proposed method will not only be limited to static paths or a limited number of movements but will be able to work efficiently in dynamic and real environments.
(6) It focuses on problem solving, not only with the distance parameter, but also taking into account the angle and direction of the movement.

The rest of this paper is organized as follows. The literature review is explained in Sect. 2. The proposed method is described with the relevant problem in Sect. 3. In Sect. 4, simulation results are presented. The final section provides conclusions and future works.

2 Literature review

In recent years, literature shows that there is interest in studies conducted on path planning and mobile autonomous vehicles [22–26], especially in three-dimensional path planning [27–30]. The general taxonomy of 3D path planning methods [18], which generally consists of four basic areas, is presented in Fig. 1. *Sampling-based algorithms* necessitate mathematical representation of information pertaining to the whole search space of the environment, known a priori. This kind of method needs some foreknown information of the whole environment (workspace). In these algorithms, the environment is represented by several nodes. They employ matching random search techniques in the path-finding mechanism to reach the appropriate solution. The matching random search techniques are then further divided into active and passive categories. In the active types, the relevant algorithm can obtain a suitable path to the destination all by its own processing procedure. Passive means algorithms only generate a path from initial to destination stations, and thus a combination of search algorithms is used to select the best feasible path in the net map where many possible paths exist. Some related case studies are presented in [31, 32]. Exploration in a *node-based optimal algorithm* is performed using a decomposed graph. These algorithms are informed search methods. These methods find an optimal path based on certain decomposition. Numerous studies have been proposed in this category [33–35]. *Mathematical model-based algorithms* include linear algorithms and optimal control techniques. These algorithms model the environment (kinematic constraints) as well as the parameters of the system (dynamic). Afterwards, they map the bounds of these two parameters based on the cost function bound. Some studies in this category are presented in [36–38]. The methods in these three categories suffer from high time complexity and local minima capture, especially where mobile robots face multiple constraints when planning a path. Metaheuristic algorithms, a subset of *nature-inspired algorithms*, are the fourth category of taxonomy that imitate biological interactive behaviours or physical events [30, 39, 40]. These methods attempt to find an almost optimal path by eliminating the process of creating complex environment models based on stochastic approaches [41, 42]. As mentioned earlier, this study proposes a hybrid path planning method. The proposed hybrid method with three versions has been inspired by sampling and metaheuristic algorithms where the main object is finding an optimal path. Mathematically based algorithms require complex calculations; on the other hand, the node-based algorithms can operate in certain and limited conditions. Therefore, methods in the other two categories generally attract more attention from researchers. Therefore, the possibility of developing new algorithms may increase. So, in this section, mainly sampling and metaheuristic-based studies are discussed.

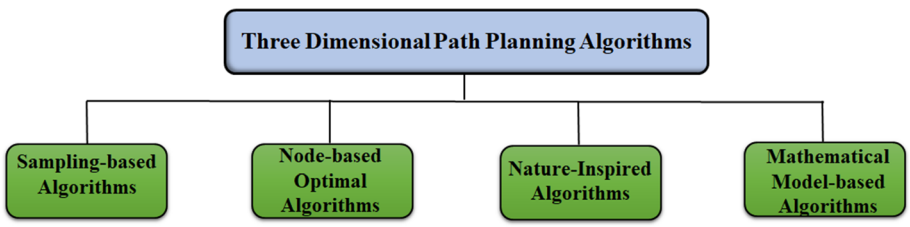


Fig. 1 3D path planning algorithms taxonomy [18]

2.1 Review on sampling-based path planning methods

In sampling-based algorithms, like many path planning methods, the map and movement areas are predetermined, but the stations in the middle are not known except for the start and end stations. This seems that these methods are dynamic and are used as a suitable solution for path planning in mobile autonomous devices such as UAV. The relevant methods based on this category employ matching random search techniques in the pathfinding mechanism to reach the appropriate solution and they are divided into active and passive categories. Active algorithms prefer to act randomly while passive algorithms suffer from dynamic barrier detection. The most efficient methods in sampling-based algorithms are those in the active category [18, 32]. The RRT is one of the most famous methods in the active model in this category. In the literature, many versions and improvements have been made regarding the RRT [19]. The RRT series solve the most problematic part of 3D path planning, i.e. holonomic and non-holonomic obstacles [43, 44]. The most common deficiency of the RRT algorithm, similar to other sampling-based methods, lies in the fact that it cannot be used to find an optimal path. The main purpose of RRT is to generate a path between source and destination points. In this algorithm, the next station is defined from the previous state behaviours. In a path planning with the RRT algorithm, a UAV has some possible states to move on. In this step, if an obstacle is located between the two points, that path is discarded, otherwise, the path without any obstacle is chosen. For the explanation of the RRT algorithm, the workspace of the UAVs is shown as p. In addition, $p_{obstacle}$ is the position of obstacles and p_{free} is the possible area without obstacles. Mathematically, $p_{free} \subset p$ and $p_{obstacles} \subset p$. In this algorithm, the maximum length between two points is important. The controlling of this length is realized by r parameter. The structure of RRT algorithms is described in the following five steps:

Step 1 Initialize the source and the destination positions along with parameter r and check the area (p_{free}). Here, $p_{source} \subset p_{free}$ and $p_{destination} \subset p_{free}$.

Step 2 Select a random station from p_{free}.

Step 3 Find the nearest station (p_{near}) from the trajectory that the mobile robot passed through before in order to randomly create stations using the Euclidean metric.

Step 4 If there is no obstacle between p_{near} and p_{random}, then p_{random} is the direction of our robot. Otherwise, ignore it and repeat this step until there is no obstacle in the path. If the distance between p_{near} and p_{random} is longer than r, then it is in the direction of p_{random} along with the r. At the end of this step add p_{random} in set T (trajectory) as new node.

Step 5 Repeat previous 4 steps until next node of a robot is $p_{destination}$.

Another method based on sampling is RRT* algorithm [45]. The RRT* is an optimized modified algorithm that aims to achieve the shortest path, whether by distance or other metrics. The basic principle of RRT* is the same as RRT, but two important additions have been made. First, RRT* records the distance each station has travelled relative to its next station. Second, RRT* is the tree rewiring. After a station (vertex) has been connected to the cheapest neighbour, the neighbours are again examined. Neighbours are checked if being rewired to the newly added vertex will make their cost decrease. If the cost does indeed decrease, the neighbour is rewired to the newly added vertex. This feature makes the path smoother. RRT* suffers from a reduction in performance. Due to examining neighbouring nodes and rewiring the graph, the implementation of RRT* needs more time to complete a single path. The convergence rate of RRT* is slow in pure exploration. Also, the successful balance between exploration and exploitation is not achieved and there may be a local optimal trap similar to other sampling-based algorithms. In [46], an algorithm (D* Lite Rapidly exploring Random Tree star (DL-RRT*)) has been proposed for path re-planning in dynamic radioactive environments. This method, whose purpose is to improve the convergence ratio, uses the grid feature of the D* algorithm. However, it has not been very successful in convergence ratio due to its reliance on a single tree version.

In [47], the authors have suggested a method to build two trees rooted at the start and goal configurations, respectively. The name given to its methods is Bidirectional RRT (B-RRT). Although their purpose is to improve the convergence ratio and avoid possible local optima traps, the computational load is quite high in the two trees that will cover the whole area. Quad-RRT [48] method has been suggested to solve this problem using the Graphics Processing Unit (GPU) threads. However, this mechanism means a high monetary cost. So it's hard to come to life in the real world. In [49] has been proposed an improved version of B-RRT that employed a greedy connect a heuristic for the connection of two-directional trees (as named B-RRT*). However, it is still not very successful in exploration. On the other hand, although it is recommended as a 3D path planning method, it is possible to use it mainly in 2D environments. This restriction is mainly valid to all B-RRT versions.

Besides the advantages of sample-based methods, the shortcoming of the sampling-based methods can be

summarized in three basic terms: (1) They suffer from computational overload and time inefficiency in optimal pathfinding. (2) They are not considered successful in finding an optimal path. (3) They are not useful in terms of convergence rate. The proposed method benefits from the advantages of sampling-based methods by eliminating their disadvantages. It is a hybrid approach that employs meta-heuristic algorithms to find collision-free optimal paths in a short time with the least process costs. This method is described in detail, along its three different variants, in Sect. 3.

2.2 Review on nature-inspired path planning methods

In [50] has been proposed an Interfered Fluid Dynamical System (IFDS) to solve the path planning problem. The authors suggested one method, which is based on mathematical-based algorithms, to detect obstacles shape and another method, which is based on nature-based algorithms, to find the optimal path for a UAV. The authors have taken the land impact on path planning as a key factor and suggested using a stream function method from their previous work. However, the stream function works in two-dimensional systems. Based on the principles of a liquid flowing around objects and the phenomenon of flow from start to end, a 3D soft path has been created using two techniques, namely analytical and numerical methods. The analytical method handles only spherical obstacles and involves less computation. The numerical method deals with all types of obstacles, but it requires more pretreatment and higher calculation cost. In the IFDS method, complex obstacles such as terrains, mountains, radar, and anti-aircraft systems are converted into cylindrical, conical, and spherical shapes. Afterward, the combined mathematical expression of these transformed shapes is fashioned to obtain 3D flow lines around a single obstacle. If there is more than one obstacle in the path, it is derived using the average of the velocity field. Next, methods such as more control force and virtual obstacle are proposed to deal with the stagnation point and overlapping of obstacles. Finally, a genetic algorithm is used to obtain an optimum 3D path of the UAV by taking the path length and flight height as subtargets. Studies demonstrate that the IFDS method gives better results than both numerical and analytical methods. Furthermore, the IFDS method has been compared with the ant optimization algorithm and results conclude that the IFDS method gives better results.

Perazzo et al. [51] articulate secure positioning and secure position verification, which are two critical problems in path planning. The related method focused on path planning for drones and gave the meaning of a secured path for an optimum path. In this study, 2D path planning is discussed, although it is not clearly stated. Their paper proposes a drone system that passes through a series of waypoints instead of a static station system. At each waypoint, the drone acts as a station and determines positions safely. Here the task is to determine the capability of the drone in finding the correct path for all devices. The authors state three methods called LocalizerBee, VerifierBee, and PreciseVerifierBee. In LocalizerBee method, isosceles triangles have been created to completely cover the rectangular area determined in the waypoint grid construction structure. After that, drones combine all waypoints by Waypoint Ordering, to find the shortest path with the Travelling Salesman Problem algorithm. In VerifierBee method, authors use prior knowledge of the position of the nodes in three phases. A number of waypoints are created during the initial path construction phase. The home waypoint plus three waypoints for each node are placed in fixed positions to form a minimal verifiable triangle. The Iterative Improvement phase creates the first path. The VerifierBee rediscovers the path created in Iterative Improvement phase, using a local heuristic search algorithm. VerifierBee then analyses each possible change (moving a waypoint to another location) at each step and selects the most appropriate as well as the shortest path. This process ends when a local minimum is found. There are two types of changes in the Waypoint Reordering phase. One that changes the position of the waypoint and the other that removes the waypoint and replaces it with another available way. The PreciseVerifierBee method is an extension of the VerifierBee method, where the structure of the Verifier remains the same except Initial Path Construction and Iterative improvement stages. Unlike the VerifierBee method, the angle of the drone to the ground plays an important role in the PreciseVerifierBee method. As a result of the application, VerifierBee always produces shorter paths in comparison with PreciesVerifierBee method. With an increase in the number of nodes, PrecieseVerifierBee shows worse results compared to VerifierBee or even LocalizerBee. The results of their experiments demonstrate that the proposed algorithms securely localize all the devices, even in presence of drone control errors.

Wang et al. [52] aim at improving the flight path by planning a 3D UAV route along with various types of restrictions mainly in complex combat environments. They have proposed the Improved Bat Algorithm (IBA), a mixture of Differential Evolution (DE), and the Bat Algorithm (BA). DE is an evolutionary algorithm that creates new candidate solutions by uniting many other members in the same population as the parents. BA is an optimization algorithm inspired by the direction-finding behaviour and distance of an object by taking advantage of the echo of a bat, the echolocation. In order to improve the

selection of a bat during the IBA location update process, a DE mutation factor was employed. The basic idea behind the IBA is that the bat group uses ecological placement to detect distance and random flight over the search area, updating its location and speed. The IBA bat flight aims to find its food/prey (best solution, most convenient way). The authors use the B-Spline Curve Correction Strategy to further improve the resulting path. It is shown in [52] that their algorithm, IBA, demonstrates better results than BA and is an effective tool that can be used in UAV path planning problems.

In [22] has been presented a new method for a mobile robot in an uncertain obstacle-based environment based on Firefly Algorithm (FA). It solves the challenges of navigation by avoiding the random movement of fireflies and minimizing the computational cost. The authors defined an objective function, which is controlled by the trial and error method. With this function, the paths to be chosen are decided. In this study, which aim is crowded environments, the choice of the obstacle closest to each station is required for path planning and an equation has been defined for this purpose. However, this equation, which contains very few parameters, may not be successful in real and complex environmental conditions. Moreover, this study focused solely on 2D path planning. In [29] has been proposed ground robot based on a new version of Ant Colony Optimization (ACO) for 3D path planning. It defines a new phenomenon update mechanism and constructs various paths between the initial and target points of a robot. It avoids obstacles and is used for solving the easily falling into local optimum and long search times in 3D path planning problems. Performance analysis could not be comprehensive by comparing this proposed method with only the basic ACO method. In addition, generally, it has a higher time and space complexity than GWO-based methods, this is due to the nature of ACO.

The GWO algorithm may be more likely to be successful than other metaheuristic methods in this type of problem on various parameters due to its working mechanism. One of the most important and effective of last studies to solve 3D path planning problem is based on the GWO algorithm, which has been presented in [40]. The authors have proposed a new 3D path planning method for UAVs and they used the GWO algorithm to decrease the complexity in finding paths to find a minimal cost best path. The GWO algorithm is based on the hunting behaviour of grey wolves. Grey wolves are classified as alpha, beta, delta, and omega, where each type of wolf exhibits distinct functionally among encircling, attacking, and hunting functions. In [40] there are three different maps with a varying number of obstacles. Furthermore, there are five UAVs with different initial and final stations. According to the results of the study, the better path cost with best time complexity in the GWO-based method in comparison with other methods such as Dijkstra, A*, D*, and a few other famous metaheuristic-based methods is obtained. Euclidean distance between stations, visited by UAVs, is used to calculate the cost of the path. They claim that paths generated using the proposed method have low path calculation costs. Most of the obstacles in this study have been located in the center of the maps that the UAVs not more effort to reach the destination. According to this paper, GWO-based methods, with the specific features and advantage of the nature of their algorithm, perform a more balanced and better performance in similar problems. Therefore, GWO-based algorithms are sought after in many research and application areas due to their balanced behaviour among various metaheuristic algorithms. For this reason, we use three variants of GWO to propose our path planning method for obstacle detection and avoidance, random movement avoidance, and optimal pathfinding. By applying each of these three GWO-based algorithms separately in our method, three different methods are developed to solve the 3D path planning problem.

In addition, the general shortcomings of metaheuristic methods can be outlined as follows: (1) They use fixed number of intermediate nodes with predetermined locations. Therefore, they can use only a limited number of intermediate nodes. (2) They do not deal with the angle and direction of motion and only cater to simple parameters such as the distance to travel. Therefore, in this study, a novel hybrid method is proposed to solve the related problems by providing all the advantages of each category. This method has a working mechanism that finds the optimal paths without collisions in the shortest possible time with the least process cost. The performances of each solution are discussed and analysed in chapter 4. Also, their performance is compared with several other methods. For this, sampling and metaheuristic-based hybrid mechanisms are suggested. Therefore, in this paper, it will be sufficient to have only the starting and ending stations for each UAV, and it will find the best choices to go along the path.

3 Proposed novel algorithms

As mentioned before, one of the most significant and contemporary problems in robotics is 3D path planning for mobile robots. It is necessary to find the optimal (or close to optimal) path between two source and destination points for robots to collision-free move without human intervention. In this paper, a new 3D path planning method is presented using three metaheuristic algorithms for autonomous UAVs. In the metaheuristic-based path planning methods, the selectable stations between the start and endpoints of each mobile autonomous robot (e.g. UAV)

may be known before and therefore routes can be static or each robot may have a limited number of directions of movement (especially, in discrete and grid areas) and they are shortcoming. In other words, the stations between the target and start points of each robot are pre-determined and the positions of these stations (next candidate stations) are also fixed. Therefore, most of the metaheuristic-based methods used in solving such problems are not concerned with the angle and direction of movement, but only with the distance to travel or other possible parameters. Also, as mentioned earlier, they are not good at using memory. It is worth noting that this working mechanism is not also exclusive to metaheuristic algorithms, whereas, in this paper, the direction, as well as the path length, has an effect on the next stations' selection. Also, the number and location of the stations between the two starting and final points are uncertain. In this study, the next station for each robot is unclear, and the intermediate stations are not fixed and predetermined. In this way, it can be applied in more realistic environments and applications. On the other hand, the proposed method can solve the problems of sampling-based algorithms. In the RRT and all its variances, the intermediate stations are chosen randomly (unlike metaheuristic methods). Therefore, they are often fast and also useful in real applications. However, these have not been very successful in finding optimal paths. In addition, they suffer from computational overload and time inefficiency in optimal pathfinding. Therefore, in this paper, a hybrid method with three versions inspired by the two categories (sampling and metaheuristic-based algorithms) are recommended. In this way, both the intermediate stations do not need to be determined beforehand and will obtainable optimized paths by avoiding completely random movements. In this paper, an improvement on the RRT algorithm, namely Adapted-RRT, is presented and it is applied with three different well-known metaheuristic algorithms; GWO, I-GWO, and Ex-GWO. These methods are named Adapted-RRT_{GWO}, Adapted-$\text{RRT}_{\text{I-GWO}}$, and Adapted-$\text{RRT}_{\text{Ex-GWO.}}$

3.1 Definitions

The path planning problem is related to finding optimal paths in a defined environment containing various obstacles. UAVs detect these obstacles and attempt to find a path without collisions. UAVs can be categorized under the mobile robots that meant to fly from one point to another autonomously in the most complicated three-dimensional (R^3) environments. In these environments, UAVs must consider the Z dimension alongside the X and the Y dimensions. In some cases, environments that UAVs are considered to operate inside can be defined as the workspace, w. The most difficult part of complicated environments for UAVs is the various shapes of obstacles. The ith obstacle in the workspace can be represented aswo_i. All the free parts of the environment are the set of coordinates,w_{free}. So, free spaces in the environment that the mobile robots can move inside without facing any possible collision can be calculated using$w_{\text{free}} = wU_i wo_i$. Both the initial (source, start) point of the mobile robot,w_{source}, and it's final (destination, target) point in the workspace, $w_{\text{destination}}$ should be the element included in thew_{free}. The definition of 3D path planning can be defined as$(S) = w_{\text{source}}$, and$(T) = w_{\text{destination}}$. Here δ represents the functions of initial and final points of a robot, respectively, and the bounded variation is given as: $[S, T] \rightarrow R^3$. If there is any continues process Φ exists that can attain$\delta(\tau) \in w_{\text{free}}$, then this process can be called as 3D path planning. Here τ is a set in which positions are located on the path that takes the UAV from source to destination. The main purpose of 3D path planning is to find a trajectory between the source (start, initial) and the destination (final, target). For that reason, a path planning function is defined in Eq. 1.

$$f(\text{source}, \text{destination}) \rightarrow \text{Trajectory} \tag{1}$$

where source and destination denote relative coordinates of the source $(X_{\text{source}}, Y_{\text{source}}, Z_{\text{source}})$ and the destination $(X_{\text{destination}}, Y_{\text{destination}}, Z_{\text{destination}})$ positions on the map. Each path contains a cost for motion from source to destination. There are different parameters that define cost between the two points. In each map, the positions of n number of search agents in the m dimensional search space can be defined as the matrix in Eq. 2.

$$\text{positions} = \begin{bmatrix} p_1^1 & p_1^2 & \cdots & \cdots & p_1^m \\ p_2^1 & p_2^2 & \cdots & \cdots & p_2^m \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_n^1 & p_n^2 & \cdots & \cdots & p_n^m \end{bmatrix} \tag{2}$$

where p^m's represents position coordinates of each station that the mobile robot takes on the map. In most studies, cost is calculated using consumption of energy, Euclidean distance, and velocity. In this study optimal cost is considered in a path with minimum Euclidean distance between two points. Due to the Euclidean metrics system distance between 2 completely randomly selected points in the three-dimensional workspace can be expressed as: $D(\text{station}_1, \text{station}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$. For the calculation of the cost (length of the trajectory) of n^{th} UAV can be transformed into the sum of the Euclidean distances between tuples from source to the destination stations, we present the equation as below.

$$\text{Cost} = \sum_{m=S}^{D} \sqrt{(x_{m+1} - x_m)^2 + (y_{m+1} - y_m)^2 + (z_{m+1} - z_m)^2} \tag{3}$$

Path planning itself for the mobile robots sometimes may not be enough to solve the problem. The 3D path planning algorithm should find the optimum paths to save time, energy, or other consumable sources for mobile robots. The definition of optimal path planning can be defined as follows. Given a path planning problem function in Eq. 1 and cost function in Eq. 3, the definition of the path planning is fulfilled and is found a path δ'. Here, Cost(δ') is equal to minimum Cost(δ) and δ is the set of all possible paths. In this situation, δ' is an optimal path, Φ' is the optimal path planning.

Figure 2 shows a sample trajectory between the source and destination stations in 2D and 3D perspectives. In the trajectory, there are some stations that the UAVs movement over stations. The S indicates the initial station, and the D is the final station. The p presents the possible stations that UAVs can move in the environment. In all path planning methods, the next station is selected from each current station and this process continues until it reaches the destination or target station. While in some methods the stations' elections are chosen randomly, in others, they are realized by a certain mechanism. In the final phase of all methods, the sum of all tuples costs based on certain mechanisms, that have been calculated, is obtained; giving the cost of the path. In the proposed method in this study, the cost of each tuple on the path is calculated by a fitness function (Eq. 3). Therefore, it is possible to find an optimum or close to an optimal path with a minimum cost between two points.

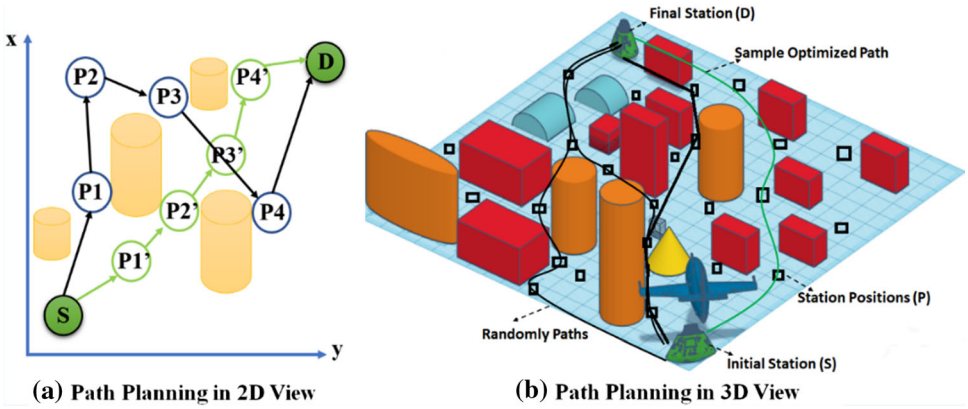
The other term that needs to be defined is random position matrix. Hence, primarily, the proposed method initializes the random position matrix. Each row of position matrix defines the path, and the columns represent the number of stations, in the path, to the destination. The $\left(x_n^m, y_n^m, z_n^m\right)$ presents coordinates of each station where m is the aforementioned index of stations and n is the number of search agents in each method (Table 1). The search agents are the configuration parameter of the metaheuristic algorithms. Then, for each metaheuristic algorithm, a search space, based on the position matrix, is initialized. Table 2 represents the distance between tuples. In this table, each row represents a path length. Each element of the row shows the distance between two points as $d^n_{(i,j)}$, where i is the current state and j is the previous state.

3.2 Map design

Typically, the first step in path planning is to represent the workspace as a map. The presence of obstacles in the maps makes the task of finding an optimal path a bit complex for the UAVs. The challenge is avoiding the obstacles and reaching the final position with minimum costs. In this paper, four maps have been considered to evaluate the proposed method; a small map, a medium map, a crowded medium map, and a large map. In Table 3, the boundaries for the four maps are presented. Furthermore, three UAVs with dissimilar start and end positions are used. These three-dimensional points of UAVs are listed in Table 4. The number of obstacles in each map is different, and the position of each obstacle is depicted in Table 5. In this study, obstacles are assumed to be three-dimensional quadrilaterals of different sizes.

3.3 Obstacle management

Obstacle avoidance is one of the many challenges that exist in the path planning problem. In this study, a mechanism is proposed to avoid the collision of the UAVs with obstacles, which benefits from geometric and calculus-based formulae. The proposed obstacle avoidance mechanism has an important role in finding the shortest path without collision. This mechanism entails two main steps that take place in a sequential fashion.



(a) Path Planning in 2D View (b) Path Planning in 3D View

Fig. 2 The randomly and optimized trajectory

Table 1 The position matrix of each path

Path	1	2	⋯	m
1	(x_1^1, y_1^1, z_1^1)	(x_1^2, y_1^2, z_1^2)	⋯	(x_1^m, y_1^m, z_1^m)
2	(x_2^1, y_2^1, z_2^1)	(x_2^2, y_2^2, z_2^2)	⋯	(x_2^m, y_2^m, z_2^m)
⋮	⋮	⋮	⋮	⋮
n	(x_n^1, y_n^1, z_n^1)	(x_n^2, y_n^2, z_n^2)	⋯	(x_n^m, y_n^m, z_n^m)

Table 2 The search space that represents distance between tuples

Path	1	2	⋯	m
1	$d^1_{(1,s)}$	$d^1_{(2,1)}$	⋯	$d^1_{(D,m)}$
2	$d^2_{(1,s)}$	$d^2_{(2,1)}$	⋯	$d^2_{(D,m)}$
⋮	⋮	⋮	⋮	⋮
n	$d^n_{(1,s)}$	$d^n_{(2,1)}$	⋯	$d^n_{(D,m)}$

Table 3 3D map boundary

Map	Start boundary	End Boundary
Small map	(0, 0, 0)	(50, 50, 50)
Medium map	(0, 0, 0)	(100, 100, 100)
Crowded medium map	(0, 0, 0)	(100, 100, 100)
Large map	(0, 0, 0)	(150, 150, 150)

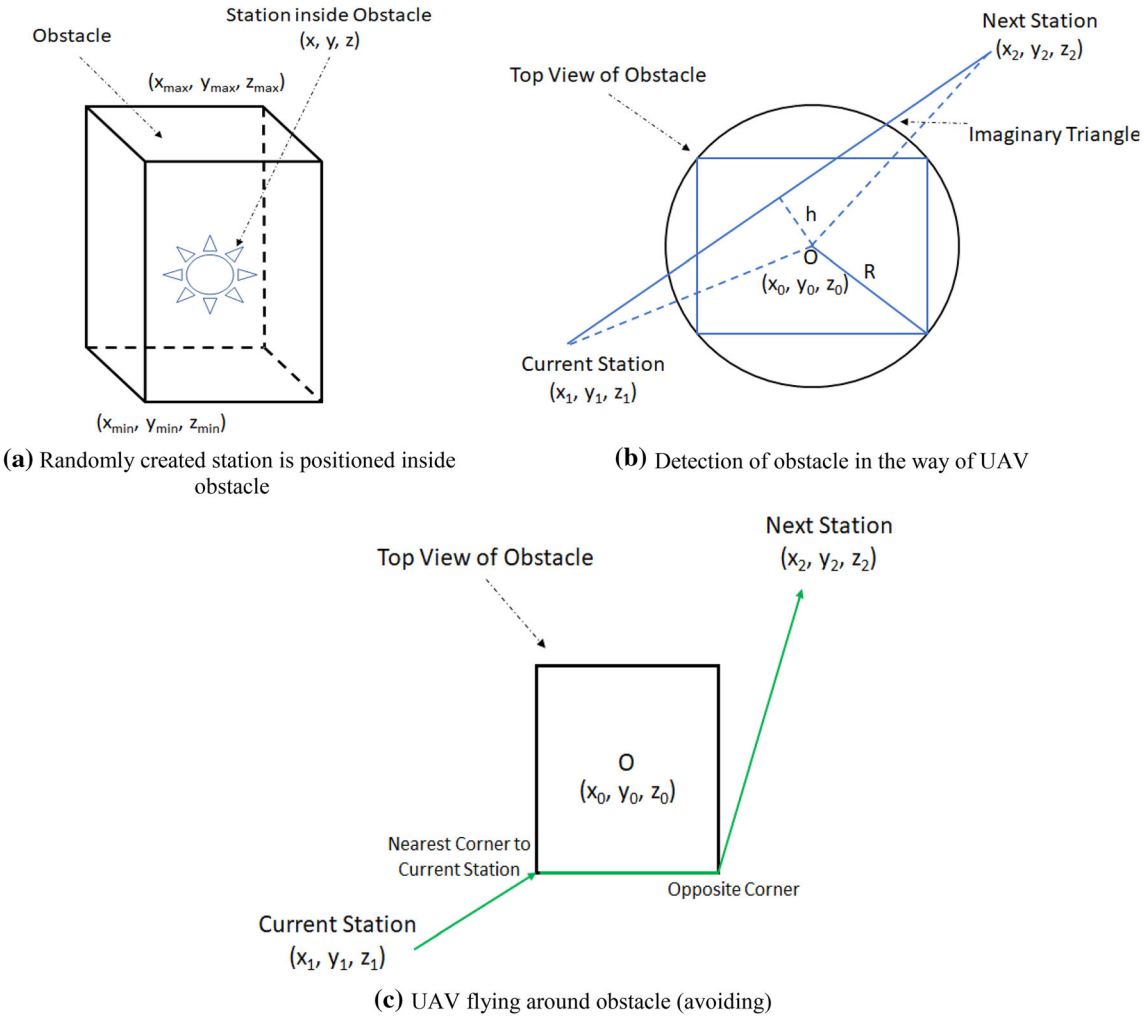
1. Whenever the randomly assigned station finds itself within the special parameters of the obstacle, it is relocated to the nearest corner of the obstacle as presented in Algorithm 1 and Fig. 3a.
2. Whenever an obstacle is encountered in the path found by the metaheuristic algorithm, the following steps are taken:
   - 2.1 Sorting Phase: sort the obstacles by Euclidean distance proximity to the current station. Significance of sorting is for detecting and avoiding various number of obstacles in series from the nearest one to the farthest one.
   - 2.2 Detection and Avoidance Phase:
     - 2.2.1 Detection: if there is a collision between the couple stations, geometric calculation method will be used. The detection mechanism is based on Algorithm 2. A virtual triangle between current, next stations, and the center point of obstacle is considered. As shown in Fig. 3b, the h is the height from center point of obstacle to the baseline, and the h can be found using Heron's formula. Besides, the R is the radius of the around circle of the obstacle. These parameters are used to check whether there is a collision or not.
     - 2.2.2 Avoidance: in this step the closest corner of the obstacle to the current station is calculated using Euclidean distance. The UAV moves to that corner and then across the opposite side of obstacle which has clear sight of next station or next obstacle as shown in Algorithm 2 and Fig. 3c.
     - 2.2.3 If the UAV does not reach to the next station, repeat phase 2.2.1 and 2.2.2.
3. If the UAV does reach the next station, repeat sorting and detection and avoidance phase for each couple in the path.

Table 4 3D map UAV initial and final positions

	Small map			Medium map		
	UAV_1	UAV_2	UAV_3	UAV_1	UAV_2	UAV_3
Initial	(0, 0, 0)	(0, 25, 0)	(0, 50, 0)	(0, 0, 0)	(0, 50, 0)	(0, 100, 0)
Final	(50, 50, 50)	(50, 25, 50)	(50, 0, 50)	(100, 100, 100)	(100, 50 100)	(100, 0, 100)
	Crowded medium map			Large map		
	UAV_1	UAV_2	UAV_3	UAV_1	UAV_2	UAV_3
Initial	(0, 0, 0)	(0, 100, 0)	(0, 50, 0)	(0, 0, 0)	(0, 75, 0)	(0, 150, 0)
Final	(100, 100, 100)	(100, 0, 100)	(100, 50, 100)	(150, 150, 150)	(150, 75, 150)	(150, 0, 150)

Table 5 3D obstacles coordinate for each map

Obstacle number	Small map	Medium map	Crowded medium map	Large map
1	(5, 7.5, 4)–(10, 20, 15)	(5, 7.5, 4)–(10, 20, 15)	(5, 7.5, 4)–(10, 20, 15)	(5, 7.5, 4)–(10, 20, 15)
2	(20, 5, 10)–(44, 44, 36)	(20, 5, 10)–(44, 44, 36)	(20, 5, 10)–(44, 44, 36)	(20, 5, 10)–(44, 44, 36)
3	(8, 30, 4)–(10, 34, 28)	(8, 30, 4)–(10, 34, 28)	(0, 91, 0)–(2, 96, 100)	(1, 5, 14)–(3, 6, 16)
4	(17, 14, 17)–(19, 16, 21)	(17, 14, 17)–(19, 16, 21)	(0, 9, 0)–(2, 8, 14)	(0, 15, 1)–(7, 16, 3)
5		(22, 5, 0)–(23, 5, 100)	(8, 30, 4)–(10, 34, 28)	(0, 18, 4)–(10, 13, 6)
6		(15, 5, 0)–(16, 0, 100)	(17, 14, 17)–(19, 16, 21)	(0, 7, 2)–(4, 5, 5)
7		(19, 1, 0)–(19, 6, 100)	(25, 20, 10)–(30, 22, 12.5)	(4, 0, 2)–(7, 5, 6)
8		(25, 6, 0)–(25, 8, 100)	(31, 20, 15)–(31, 20, 18.8)	(0, 15, 0)–(10, 20, 1)
9			(42, 42, 42)–(42, 42, 44)	(7, 8, 9)–(10, 11, 12)
10			(51, 22, 31)–(62, 31, 39)	(5, 89, 140)–(5, 90, 144)
11			(60, 15, 0)–(75, 40, 100)	(9, 15, 2)–(12, 19, 6)
12			(85, 85, 0)–(90, 90, 100)	(85, 15, 23)–(86, 16, 26)



(a) Randomly created station is positioned inside obstacle

(b) Detection of obstacle in the way of UAV

(c) UAV flying around obstacle (avoiding)

Fig. 3 Obstacle avoidance mechanism

Algorithm 1. Check the station position

```
If (x < xmax && y < ymax && z < zmax)
    If (x > xmin && y > ymin && z > zmin)
        //Station is positioned inside the obstacle
        Find nearest corner to the station and reassign position
    End
End
```

Algorithm 2. Collision detection and avoidance

```
If (h <= R)
    //Collision occurs
    UAV finds nearest corner to current station, across the opposite
    corner of the obstacle, from that corner UAV pass to the
    next station
Else
    //Collision does not occur
End
```

3.4 Grey wolf algorithms: GWO, I-GWO, and Ex-GWO

In this section, the GWO-based algorithms used in our proposed method are summarized. The GWO algorithm is inspired by grey wolves in nature and the natural behaviour of the grey wolves are mathematically modelled. The main behavioural traits of the wolves are encircling, hunting, and attacking the prey. There are four types of wolves in each pack; alpha, beta, delta, and omega. Each wolf has different responsibilities in the pack. Alpha, beta, and delta wolves encircle the prey, whereas the omega wolves follow the first three wolves' position. In simple terms, omega wolves update their own position based on the other wolves in order to attack the prey. The GWO algorithm is mathematically defined using Eqs. 4–11. Here t indicates the current iteration, T demonstrates maximum number of iterations, $\vec{X}$ indicates the position vector of a wolf. Also, D is a vector that depends on the location of the target. Here the coefficient vectors $\vec{A}$ and $\vec{C}$ are considered to lead in encircling the prey (Eqs. 6 and 7). These parameters control the trade-off between the exploration and exploitation phases in all variants of used GWO algorithms. In addition, $\vec{a}$ is linearly decreased from 2 to 0 over the courses of iteration. It is used to get closer to the solution range and r_1 and r_2 are the random vectors in a range of [0, 1]. In all variants, when $\vec{A}$ is less than 1, the wolves in the pack attack to hunt, otherwise, they try to find prey to be hunted. Figure 4 shows the working mechanisms of these algorithms by considering exploration and exploitation phases.

$$\vec{D} = \left|\vec{C} \cdot \vec{X_p}(t) - \vec{X}(t)\right| \tag{4}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{5}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a} \tag{6}$$

$$\vec{C} = 2 \cdot \vec{r_2} \tag{7}$$

$$\vec{a} = 2\left(1 - \frac{t}{T}\right) \tag{8}$$

$$\begin{aligned}\vec{D_\alpha} &= \left|\vec{C_1} \cdot \vec{X_\alpha} - \vec{X}\right|, \vec{D_\beta} = \left|\vec{C_2} \cdot \vec{X_\beta} - \vec{X}\right|, \vec{D_\delta} \\ &= \left|\vec{C_3} \cdot \vec{X_\delta} - \vec{X}\right|\end{aligned} \tag{9}$$

$$\begin{aligned}\vec{X_1} &= \vec{X}_a - \vec{A_1} \cdot \vec{D_\alpha}, \vec{X_2} = \vec{X}_\beta - \vec{A_2} \cdot \vec{D_\beta}, \vec{X_3} \\ &= \vec{X}_\delta - \vec{A_3} \cdot \vec{D_\delta}\end{aligned} \tag{10}$$

$$\vec{X}(t+1) = \frac{\vec{X_1}(t) + \vec{X_2}(t) + \vec{X_3}(t)}{3} \tag{11}$$

In I-GWO algorithm, the hunting mechanism of the GWO has been modified. The authors of I-GWO algorithm supposed that the alpha wolf has the best knowledge about the prey. The second wolf in the pack follows the alpha wolf, and in the same way, the nth wolf in the pack follows all $n-1$ wolves before it in the pack. Simply putting, the alpha wolf guides the other wolves in the pack for hunting. There are also some modifications in I-GWO mathematically from the classic GWO. In the I-GWO, each grey wolf in the pack encircles the prey; mathematically presented in Eqs. 4 and 5. In this method, authors have been used a new version of the $\vec{a}$ parameter so as to improve the convergence speed of the metaheuristics, which is defined in Eq. 12. The I-GWO benefits from the position of the alpha wolf while updating other wolves' positions in the pack (Eqs. 13–15).

$$\vec{a} = 2\left(1 - \frac{t^2}{T^2}\right) \tag{12}$$

$$\vec{D_\alpha} = \left|\vec{C_\alpha} \cdot \vec{X_\alpha} - \vec{X}\right| \tag{13}$$

$$\vec{X_1} = \vec{X}_\alpha - \vec{A_1} \cdot \vec{D_\alpha} \tag{14}$$

$$\vec{X_n}(t+1) = \frac{1}{n-1}\sum_{i=1}^{n-1} X_i(t); n = 2, 3, \ldots, m \tag{15}$$

Ex-GWO algorithm is yet another version of the GWO algorithm. In the Ex-GWO, the first three wolves; alpha, beta, and delta wolves, all have the best knowledge about the prey. The fourth wolf in the pack, just like in the GWO algorithm, updates its own position based on alpha, beta, and delta wolves. The fifth wolf updates its own position using the positions of the first three wolves and the fourth wolf. As such, the nth wolf updates its own position based

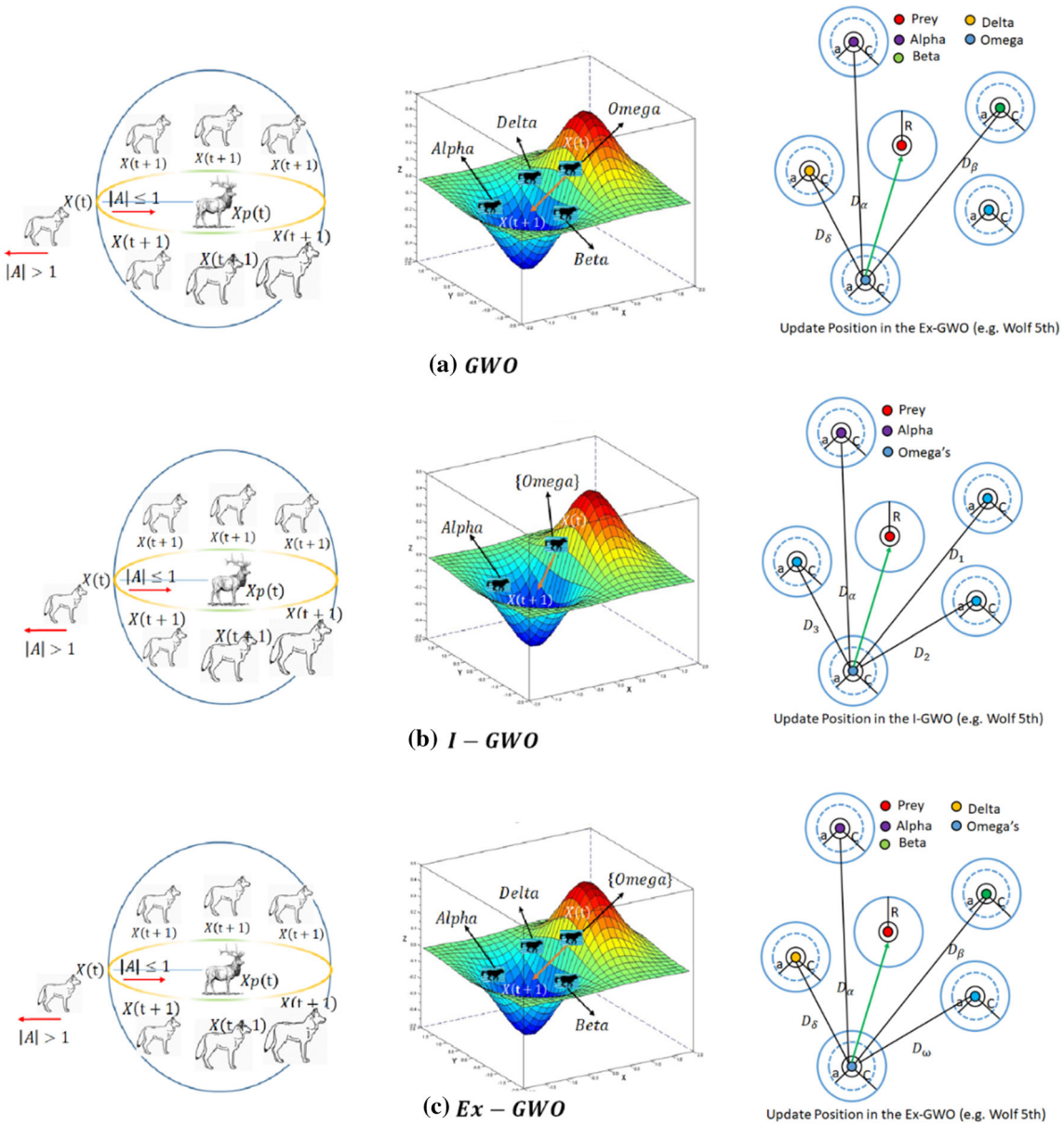


Fig. 4 Working mechanism considering exploration and exploitation **a** GWO, **b** I-GWO, **c** Ex-GWO

on the first three wolves in the pack and the $n-3$ wolves before it. So, the wolves benefit more from the packs' knowledge in order to hunt and attack the prey. In the Ex-GWO, encircling of the prey is mathematically expressed using Eqs. 4–7. The hunting and position updates are also given by Eqs. 16–18.

$$\begin{aligned}\overrightarrow{D_1} &= \left|\overrightarrow{C_1}\cdot\overrightarrow{X_1}-\overrightarrow{X}\right|, \overrightarrow{D_2}=\left|\overrightarrow{C_2}\cdot\overrightarrow{X_2}-\overrightarrow{X}\right|, \overrightarrow{D_3}\\ &= \left|\overrightarrow{C_3}\cdot\overrightarrow{X_3}-\overrightarrow{X}\right|\end{aligned} \tag{16}$$

$$\begin{aligned}\overrightarrow{X_1} &= \overrightarrow{X}_1-\overrightarrow{A_1}\cdot\overrightarrow{D_1}, \overrightarrow{X_2}=\overrightarrow{X}_2-\overrightarrow{A_2}\cdot\overrightarrow{D_2}, \overrightarrow{X_3}\\ &= \overrightarrow{X}_3-\overrightarrow{A_3}\cdot\overrightarrow{D_3}\end{aligned} \tag{17}$$

$$\overrightarrow{X_n}(t+1)=\frac{1}{n-1}\sum_{i=1}^{n-1}X_i(t); n=4,5,\ldots,m \tag{18}$$

3.5 Hybrid path planning methods: adapted-RRT$_{GWO}$, adapted-RRT$_{I\text{-}GWO}$, and adapted-RRT$_{Ex\text{-}GWO}$

In this paper, a new hybrid path planning method with three versions is proposed to use the advantages of

metaheuristic and sampling-based methods in one place. We propose an improvement in the RRT algorithm using three metaheuristic algorithms (GWO, I-GWO, and Ex-GWO). In order to use the RRT concept together with metaheuristic methods, it is necessary to make the classical RRT method compatible and efficient. Thus, the success rate of the proposed method in finding optimal solutions may be increased. Based on this, a newer version of the RRT method (Adapted-RRT) is recommended in this study. Figure 5 shows the working mechanism of the RRT and Adapted-RRT.

The random selection of the intermediate stations in all RRT methods is realized by a mechanism with considering possible obstacles. In the RRT, if an obstacle is located between the two points, the relevant path is discarded, otherwise, the path without any obstacle is chosen. However, this algorithm or its related improvements, either failed to find the optimal path or they suffered over-computed processes to find the optimal path and were not efficient in time. As stated in Sect. 2.1, the terms defined for this algorithm are used. As shown in Fig. 5a, in the RRT, when the UAV is in state *1*, there are three possible stations, a UAV chooses a station without obstacle in the path and follows this path to reach the destination.

Adapted-RRT avoids possible obstacles without making random movements. It is a hybrid method that uses metaheuristic algorithms in its working mechanism to find collision-free optimal paths in a short time with the least process costs. It has also suitable behaviour in convergence rate. We define the *r* parameter more efficiently with a different mechanism and also use this value in choosing the appropriate path length in a next station selection of each UAV. For this, it acts with the *X* parameter in its metaheuristic algorithms. It also tries to find the optimal direction to reach the destination station via a shorter route as collision-free. For this, the angle is defined and optimized by metaheuristic algorithms. In the Adapted-RRT is

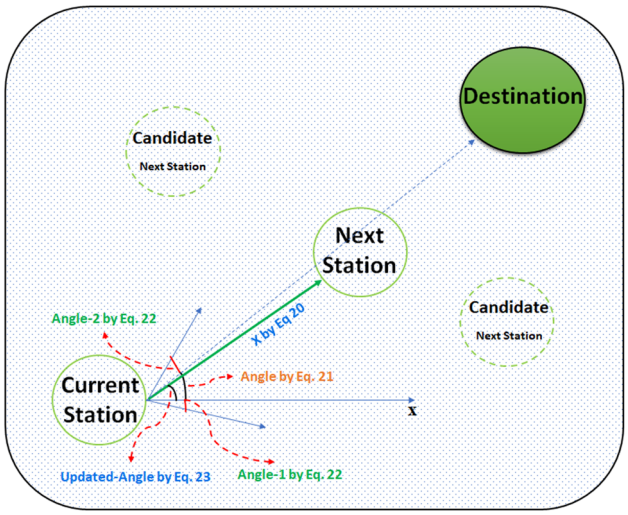


Fig. 6 Angle finding and movement length updating mechanism in a certain iteration of proposed method- An example

deployed two important operations to select next stations and finally generate path; the *length of each movement* and the *direction of each movement*. For this, three metaheuristic algorithms are used to perform these two operations in the most appropriate way. The GWO algorithms may be the best choice to perform these operations in a balanced way. Accordingly, three different variants of this algorithm are used. The Ex-GWO-based path planning method may be performed more successfully in larger and crowded environments with more populations and iterations and I-GWO-based path planning method may give good results in medium and smaller with less populated environments. In addition, the GWO-based method is generally between these two methods and behaves in a balanced way. Equation 3 is used to calculate the total path cost of the selected stations in all versions of our proposed method. The working mechanism of the proposed method,

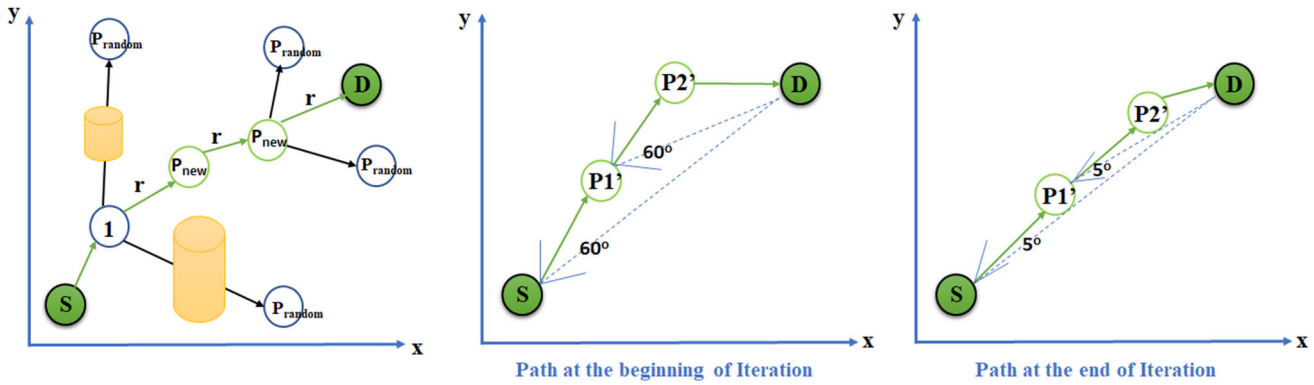


Fig. 5 Working mechanism sample of **a** RRT and **b** Adapted-RRT

which is based on sampling and metaheuristic algorithms, is illustrated schematically in Fig. 5b.

3.5.1 Calculation of length of each movement

In the proposed method, generally, each UAV has a travel length of 0 to r. This value indicates the length of each movement to a next station (new station). As mentioned before, the Euclidean distance determines the cost of each path. There is a limitation parameter that forces the UAVs to move around to the r value. It is necessary to use this since the next stations are not initially defined, otherwise, the agents reach the destination or a random station without any restriction. The r value calculation in the Adapted-RRT is obtained using Eq. 19. Here *dim* represents stations number between source and destination states. On the other hand, in this study, length of each movement, which is selected using metaheuristic algorithms, is aimed to be the best length. The value calculated from metaheuristic algorithms is named with X. Therefore, firstly, an X value is calculated from the GWO, I-GWO, and Ex-GWO algorithms based on Eqs. 11, 15, and 18, respectively. If this value is greater than r, the length of the value is calculated according to the first part of Eq. 20, otherwise, it is obtained by another part of Eq. 20.

$$r = (p_{\text{destination}} - p_{\text{source}})/(\dim + 1) \tag{19}$$

$$\begin{cases} X = r + (X - r) * \text{rand}(0,1) & \text{if } X > r \\ X = X + (r - X) * \text{rand}(0,1) & \text{if } X \leq r \end{cases} \tag{20}$$

3.5.2 Conclusion of direction of each movement

In this method, another control parameter to find an optimal path is direction. The movement direction is not declared, because it is desired to avoid standing in the current station. Thanks to this operation, the most suitable next station for each current station is selected and therefore the corresponding UAV moves to the selected next station position. In this step, the Adapted-RRT algorithm still uses three GWO algorithms to find an optimal angle to determine the movement direction. The angles obtained from these metaheuristic algorithms are updated to a more precise value as the iterations progress. The parameter a is very important to determine the most appropriate angle. While a has the largest value, the larger angle is chosen and as the iterations progress, a value becomes smaller and hence the angle will also narrow. The a is linearly decreased from 2 to 0 over the courses of iteration. This parameter helps the Adapted-RRT algorithm in finding a better solution over the iterations of the proposed method. As mentioned before, the first process in the work of the GWO algorithms is to initialize a, A and C values (Eqs. 6, 7, 8, and 12). The coefficient vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are considered to lead in encircling their hunt (angle) based on Eqs. 4 and 5. These parameters control the trade-off between the exploration and exploitation phase in each GWO algorithms.

In angle calculation, first, the angle between current and destination stations is calculated by Eq. 21. For this, the angle is determined with cosines. Here, the Euclidian distance between current and destination stations is effective in finding the angular value. The parameter *EuclDist* represents this distance. Then, Eqs. 22 and 23 are used to find the angle between current and next stations. It is most important in the next station selection. The parameter β determines the movable degrees that take UAVs to the destination. This parameter is experimental that in this study it is chosen 15 degrees. Equation 24 is used to find the coordinates of candidate state based on the angle and X value obtained. Since the problem is three-dimensional, there are two angles, so it is quite natural to update two axes. The working mechanism of the angle finding and updating is shown in Fig. 6 with a simple example.

$$cos(\text{angle}) = \frac{X_{\text{destination}} - X_{\text{current}}}{\text{EuclDist}_{\text{destination,current}}} \tag{21}$$

$$\begin{aligned} \text{Angle}_1 &= (\arccos(\text{angle}) + (\beta * a)), \\ \text{Angle}_2 &= (\arccos(\text{angle}) - (\beta * a)) \end{aligned} \tag{22}$$

$$\text{UpdatedAngle} := \text{rand}(\text{Angle}_1, \text{Angle}_2) \tag{23}$$

$$\begin{aligned} &X_{\text{next}} = X_{\text{current}} + X * cos(\text{UpdatedAngle}), \\ &Y_{\text{next}} = Y_{\text{current}} + X * sin(\text{UpdatedAngle}), Z_{\text{next}} = Z_{\text{current}} \end{aligned} \tag{24}$$

Based on these two operations, the most suitable next station is selected for each current station. Therefore, the optimal path will be determined for each UAV. The pseudocode and flowchart of the proposed method with three variants of metaheuristic algorithms (Adapted-RRT_{GWO}, Adapted-$\text{RRT}_{\text{I-GWO}}$, and Adapted-$\text{RRT}_{\text{EX-GWO}}$) can be found in Algorithm 3 and Fig. 7. The general architecture of the proposed method is also represented in Fig. 8.

Algorithm 3. Pseudocode of Adapted-RRT$_{GWO,I\text{-}GWO,Ex\text{-}GWO}$

```
Initialize grey wolf population Xi (i=1, 2, …..,n )
Initialize r, a, A and C                    //Eq. 19, 12, 6, and 7
Initialize Positions matrix
Calculate fitness of each agent   //Eq.3
Xα = best search agent
While (t< Max number of iterations)
   For each search agent
      Find distance between r and solution obtained by metaheuristics
         // For Adapted-RRT_GWO :Eq. 11, For Adapted- RRT_I-GWO : Eq. 15, For Adapted-RRT_Ex-GWO :Eq. 18
      Find best length of movement   // Eq. 19 and 20
      Find best angle degree as direction of movement    // Eq. 22, 23
      Update coordinates of current search agent    //Eq. 24
   End For
Update a, A and C
Calculate fitness of all search agents
Update Xα
Insert Xα to best Positions matrix
t = t +1
End While
Return Xα
```

4 Simulation results

This section presents the results of the proposed method with three variants of metaheuristic algorithms (Adapted-RRT$_{GWO}$, Adapted-RRT$_{I\text{-}GWO}$, and Adapted-RRT$_{Ex\text{-}GWO}$) and evaluates their performance. They are compared with 8 mixed sampling and metaheuristic-based methods; BPIB-RRT* [49], tGSRT [53], GWO [20], I-GWO [21], Ex-GWO [21], PSO [54], Improved BA [52], and WOA [55] using same environmental conditions. The implementation and analysis presentation has been performed using Java and MATLAB. The proposed methods and other used methods to compare are simulated on a Core i7-5500U 2.4 processor with 8 GB of RAM.

4.1 Assumptions of simulation

In this study, different scenarios have been considered in various conditions such as map size, the number of obstacles and their sizes, start and target points of UAV to better analyse the performance of the proposed method. In this regard, four maps (small, medium, crowded medium, and large), presented in Table 3, with different starting and ending boundaries have been used in simulations. Furthermore, the starting and final stations of three UAVs are listed in Table 4, whereas the obstacles coordinates are given in Table 5. The shape of obstacles is supposed rectangle. All the coordinates are presented in three-dimensional space. Moreover, the assumed environment is continuous and bounded. The optimal path costs, execution time and complexity, and convergence rate analysis is explained in the following section in detail, using various population sizes and iterations numbers as given in the couple-tuples form: (25, 40), (50, 100), (100, 100). In addition, the parameter values of metaheuristic algorithms used are presented in Table 6.

4.2 Comparison and evaluation of the path costs

In this section, the Adapted-RRT$_{GWO}$, Adapted-RRT$_{I\text{-}GWO}$, and Adapted-RRT$_{Ex\text{-}GWO}$ for path planning are analysed based on cost function (Eq. 3). All the obtained cost values are in centimetres. Also, the overall time parameter refers to the average running time of each method. As metaheuristic algorithms may obtain different as well as near solutions, we run each algorithm 10 times. The best cost values are presented with different population sizes and iteration numbers (see Tables 7, 8, 9 and 10). Each version of the proposed method has three UAVs with different start and final stations. According to the results, the Adapted-RRT-based methods are moved to the next station based on the adapting current position according to the destination position. With larger dimensions of the map and the increasing number of obstacles, the proposed algorithms give better minimum cost values, when compare with other algorithms.

Among all the algorithms used in this paper, Adapted-RRT$_{EX\text{-}GWO}$ algorithm gives the best results compared to other algorithms, while Adapted-RRT$_{I\text{-}GWO}$ and I-GWO algorithms take in the second and third ranks, respectively. Table 11 presents ranking summary of each algorithm. Briefly, each path planning method runs in the three

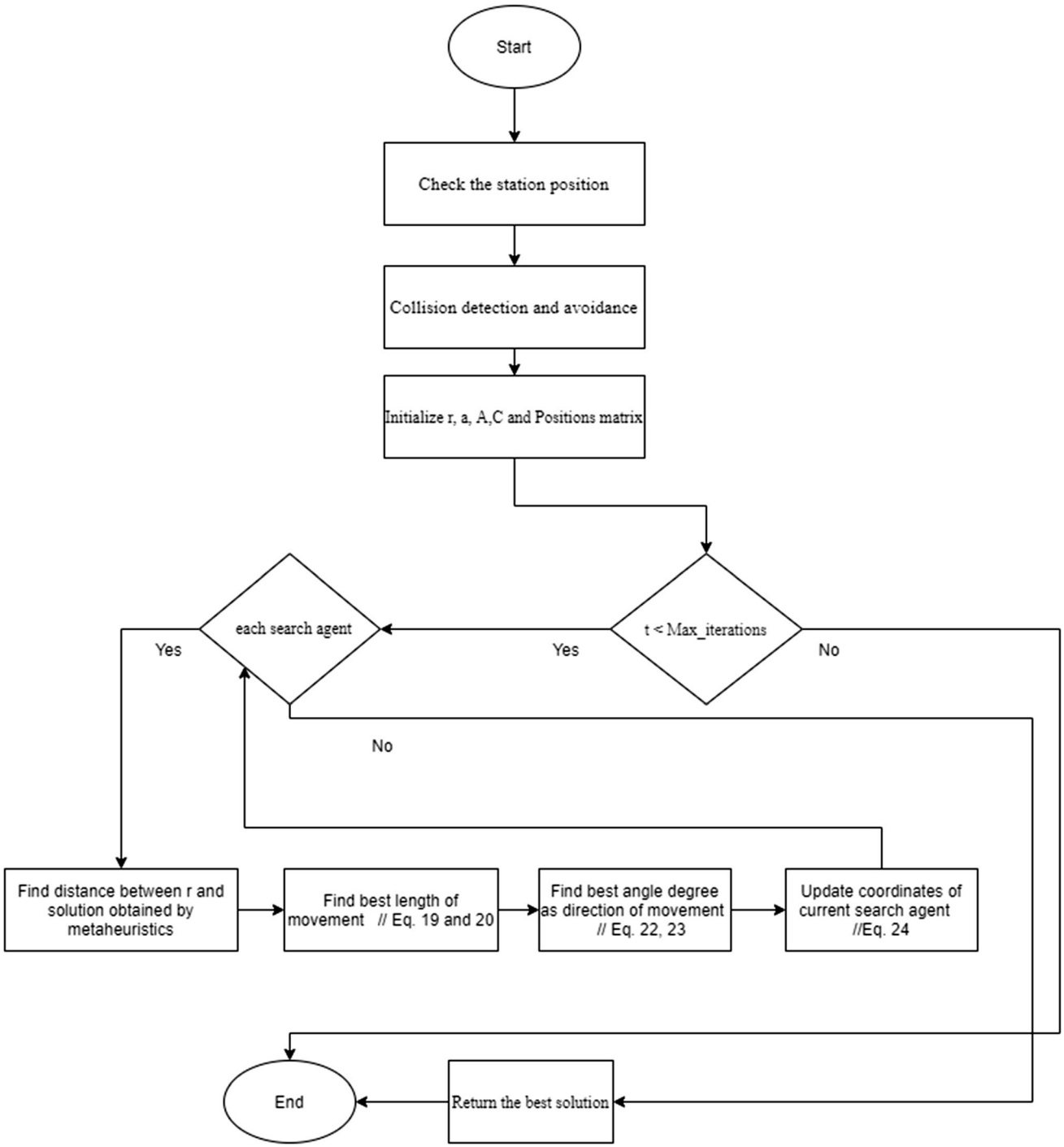


Fig. 7 The flowchart of the proposed method

different populations and iteration sizes on four maps. Table 11 shows the percentage of algorithms obtaining the minimum cost. Here, each path planning method has a different mechanism, as the metaheuristic operations and mechanism vary, the obtained results are different. The proposed adapted-RRT path planning method benefits from the parameters of the metaheuristic algorithms to find the optimal path.

Although it is not possible to make a definite generalization based on the analysis of the results obtained, a result like the following may arise. The Adapted-$RRT_{EX\text{-}GWO}$ can perform more successfully in larger and crowded environments with more obstacles. Also, the more

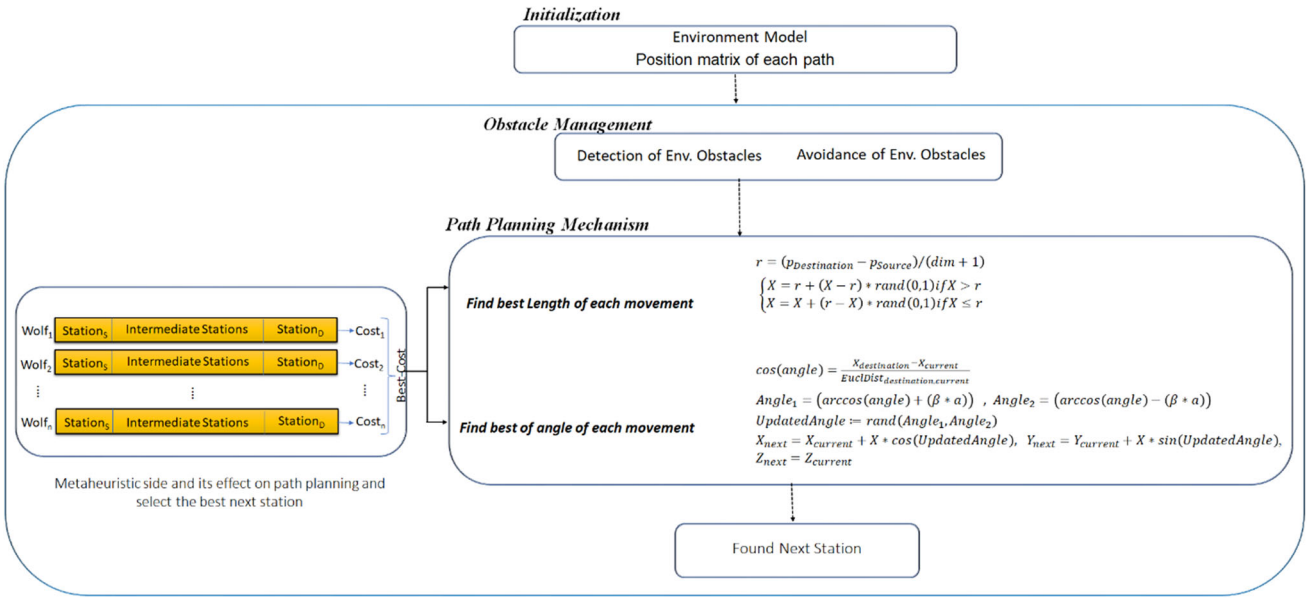


Fig. 8 General architecture of the proposed method

Table 6 The simulation parameters

Algorithm	Parameter	Value	Algorithm	Parameter	Value
GWO	a	[2, 0]	WOA	a	[2, 0]
I-GWO	A	[2, 0]		A	[2, 0]
Ex-GWO	C	2.rand (0, 1)		C	2.rand (0, 1)
				l	[− 1, 1]
IBA	Mutation scaling	0.5		b	1
	Pulse rate	0.5	PSO	Weight	0.6
	Loudness balances	0.95		c_1 and c_2 acceleration constants	1.7
	Counts of dimension to change	0.65			
	Frequency	0.5			

populations and iterations used, it may have more chance to find good results. The main reason for this is that in the Ex-GWO method, almost all wolves in the pack have an important role in each other's position update. Therefore, the wolves in the pack minimize the escape paths of the hunt (prey), and hence, the hunts can be caught faster. On the other hand, the Adapted-RRT$_{\text{I-GWO}}$ method may give good results in small-medium sized environments. In addition, it may be more successful than other GWO algorithms in fewer populations and iterations because it depends only on the alpha wolf. The GWO-based method is generally between these two methods and behaves in a balanced way.

All the paths generated for different maps have been shown in Fig. 9 is a perspective view. In this figure, it shows the path of each UAV travels in 3D. The cost of these paths was presented in the previous paragraph in Tables 7, 8, 9 and 10. In this figure, the balls show the source (initial) station of each UAV, and the star symbols show the destination state of each UAV. As mentioned before, there are three UAVs on each map, that the proposed path planning methods generate collision-free optimal paths. As such, there are three different paths generated for each algorithm; making 9 different paths for each map. The results show that the proposed methods generate optimum paths without any collision. Also, the

Table 7 Simulation results for small map

Algorithm	Pop	Iter	UAV_1 Cost				UAV_2 Cost				UAV_3 Cost				Overall Time (s)
			Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
ADAPTED-RRT_{GWO}	25	40	124	158	139	1.05E + 01	116	134	123.2	6.84E + 00	118	141	131.5	8.59E + 00	5.146
ADAPTED-$RRT_{I\text{-}GWO}$	25	40	122	155	133.4	1.27E + 01	113	151	128.7	1.25E + 01	114	151	130.7	1.24E + 01	5.454
ADAPTED-$RRT_{EX\text{-}GWO}$	25	40	123	150	137.4	9.18E + 00	110	136	124.5	9.44E + 00	114	141	129.6	9.48E + 00	5.797
GWO	25	40	110	167	142.5	2.29E + 01	102	155	123.1	2.25E + 01	115	194	162.5	2.28E + 01	4.835
I-GWO	25	40	**107**	167	137.7	1.79E + 01	**101**	149	122.4	1.56E + 01	104	179	147.2	2.62E + 01	**4.823**
EX-GWO	25	40	110	161	138	1.65E + 01	116	148	133.1	1.06E + 01	**103**	185	149.2	2.66E + 01	4.647
BPIB-RRT*	25	40	120	168	143	1.73E + 01	117	174	144.8	1.73E + 01	119	204	151.7	2.94E + 01	6.823
tGSRT	25	40	125	179	154.5	2.05E + 01	119	171	151.2	1.73E + 01	123	191	159.7	2.35E + 01	6.792
PSO	25	40	118	181	145.6	2.10E + 01	110	174	144.9	1.98E + 01	136	181	151.6	1.72E + 01	5.892
IBA	25	40	115	171	145	1.75E + 01	120	188	149.7	2.09E + 01	123	171	148.8	1.64E + 01	6.147
WOA	25	40	110	165	142.3	1.98E + 01	116	165	143	1.90E + 01	138	177	159.3	1.33E + 01	4.952
ADAPTED-RRT_{GWO}	50	100	113	147	133.5	1.06E + 01	110	131	123.2	6.75E + 00	114	135	124.4	8.77E + 00	10.862
ADAPTED-$RRT_{I\text{-}GWO}$	50	100	108	142	125.5	1.36E + 01	110	127	117	6.77E + 00	113	140	127.5	1.06E + 01	**10.861**
ADAPTED-$RRT_{EX\text{-}GWO}$	50	100	104	151	122.3	1.39E + 01	108	124	114.6	5.25E + 00	110	130	121	6.68E + 00	10.893
GWO	50	100	110	146	133.5	1.13E + 01	106	133	122.9	8.53E + 00	127	186	151.8	2.46E + 01	41.819
I-GWO	50	100	111	170	141.3	2.27E + 01	106	136	120.1	1.02E + 01	**109**	156	131.4	1.84E + 01	46.891
EX-GWO	50	100	**103**	136	118.2	1.17E + 01	107	138	123.1	1.04E + 01	117	153	134.2	1.20E + 01	42.749
BPIB-RRT*	50	100	116	174	144.2	1.96E + 01	113	181	145.2	2.21E + 01	113	181	148.2	2.26E + 01	12.358
tGSRT	50	100	115	179	153.4	2.38E + 01	114	188	154.6	2.58E + 01	114	188	156.6	2.66E + 01	12.785
PSO	50	100	105	166	133.6	1.88E + 01	**105**	161	132.3	1.67E + 01	135	169	147.3	1.35E + 01	49.147
IBA	50	100	109	162	139.6	1.91E + 01	108	162	140.6	1.91E + 01	121	165	146.9	1.45E + 01	51.148
WOA	50	100	110	160	136.9	1.79E + 01	107	155	129.6	1.54E + 01	129	177	143.2	1.79E + 01	48.159
ADAPTED-RRT_{GWO}	100	100	109	133	120	9.19E + 00	111	129	118.8	6.14E + 00	113	126	118.3	4.42E + 00	**13.239**
ADAPTED-$RRT_{I\text{-}GWO}$	100	100	106	138	123	1.06E + 01	113	123	117.9	3.25E + 00	**107**	132	119.8	9.67E + 00	17.751
ADAPTED-$RRT_{EX\text{-}GWO}$	100	100	107	132	119.2	8.11E + 00	**99**	119	111.6	6.70E + 00	108	118	111.7	5.16E + 00	19.495
GWO	100	100	**105**	134	116.3	1.21E + 01	105	132	141.5	8.55E + 00	111	154	132	1.35E + 01	144.601
I-GWO	100	100	107	153	132.5	1.92E + 01	103	137	117.7	1.00E + 01	**107**	145	126	1.44E + 01	119.672
EX-GWO	100	100	**105**	161	127.3	1.73E + 01	104	133	117.5	9.06E + 00	114	146	127.4	1.05E + 01	134.885
BPIB-RRT*	100	100	118	181	145.7	2.21E + 01	111	160	135.1	1.56E + 01	114	171	141.3	1.84E + 01	20.489
tGSRT	100	100	117	182	149.3	2.38E + 01	112	173	139.2	1.75E + 01	115	165	139.1	1.71E + 01	22.684
PSO	100	100	109	171	138.1	2.08E + 01	105	159	131.5	1.67E + 01	111	161	135.8	1.66E + 01	154.485
IBA	100	100	110	166	142	1.96E + 01	102	153	132.9	1.71E + 01	117	166	142.7	1.84E + 01	151.253
WOA	100	100	109	154	131.8	1.45E + 01	108	144	130.9	1.22E + 01	115	157	136	1.39E + 01	145.985

The bold values are the best results

Table 8 Simulation results for medium map

Algorithm	Pop	Iter	UAV_1 Cost				UAV_2 Cost				UAV_3 Cost				Overall Time (s)
			Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
ADAPTED-RRT_{GWO}	25	40	245	321	287	2.62E + 01	216	258	235.1	1.33E + 01	249	296	275.2	1.58E + 01	5.586
ADAPTED-$RRT_{I\text{-}GWO}$	25	40	255	305	282.2	1.80E + 01	202	235	213.3	1.13E + 01	**240**	289	263.4	1.73E + 01	**5.574**
ADAPTED-$RRT_{EX\text{-}GWO}$	25	40	242	302	273.8	1.97E + 01	213	261	237.1	1.69E + 01	244	280	260.3	1.27E + 01	6.205
GWO	25	40	261	347	315.3	3.12E + 01	204	263	228.2	1.95E + 01	270	327	304.7	2.17E + 01	6.241
I-GWO	25	40	255	323	295.2	2.44E + 01	**198**	283	239.7	2.63E + 01	260	324	287.5	2.76E + 01	7.605
EX-GWO	25	40	248	348	306.5	3.23E + 01	199	254	226.5	1.81E + 01	270	335	282	2.97E + 01	7.914
BPIB-RRT*	25	40	221	399	306.3	5.69E + 01	218	309	279	3.23E + 01	263	399	310.5	5.11E + 01	8.495
tGSRT	25	40	220	374	306.9	4.93E + 01	220	319	271.5	3.36E + 01	270	374	311.9	4.14E + 01	9.128
PSO	25	40	209	378	300.6	6.15E + 01	210	300	254.6	3.02E + 01	220	378	301.7	5.98E + 01	8.259
IBA	25	40	215	354	290.2	4.88E + 01	216	291	257.2	2.55E + 01	226	354	291.3	4.70E + 01	9.334
WOA	25	40	**207**	360	303.7	5.08E + 01	209	278	251.2	2.55E + 01	272	360	310.2	4.01E + 01	8.589
ADAPTED-RRT_{GWO}	50	100	242	317	278.8	2.46E + 01	190	221	203	1.01E + 01	227	243	234.2	4.98E + 00	20.838
ADAPTED-$RRT_{I\text{-}GWO}$	50	100	276	336	304.4	1.89E + 01	188	243	212.5	1.92E + 01	**225**	265	244.1	1.40E + 01	18.635
ADAPTED-$RRT_{EX\text{-}GWO}$	50	100	253	320	293.4	2.30E + 01	**180**	229	207.7	1.64E + 01	226	242	233.2	6.05E + 00	**18.713**
GWO	50	100	239	341	297.6	4.31E + 01	192	265	231.6	2.52E + 01	261	315	293.7	1.99E + 01	44.569
I-GWO	50	100	**236**	320	281.6	3.69E + 01	197	265	225.8	2.33E + 01	**225**	295	262.6	2.15E + 01	48.626
EX-GWO	50	100	242	352	308.6	3.96E + 01	201	258	231.9	2.10E + 01	237	296	271.3	1.95E + 01	44.991
BPIB-RRT*	50	100	245	404	313.4	5.50E + 01	222	357	297.4	4.14E + 01	293	344	301.8	3.07E + 01	19.148
tGSRT	50	100	251	409	332	5.37E + 01	224	348	286.7	4.04E + 01	288	340	291.9	3.16E + 01	19.785
PSO	50	100	240	378	298.4	4.32E + 01	198	328	275.1	4.46E + 01	285	329	284.5	3.37E + 01	57.145
IBA	50	100	239	381	305.9	4.64E + 01	198	333	283.2	4.01E + 01	298	339	316.1	2.30E + 01	51.148
WOA	50	100	240	351	294.8	3.73E + 01	197	301	256.6	3.67E + 01	305	342	312.4	2.19E + 01	49.254
ADAPTED-RRT_{GWO}	100	100	242	314	277.5	2.38E + 01	187	221	203.7	1.26E + 01	218	232	224.5	4.86E + 00	49.035
ADAPTED-$RRT_{I\text{-}GWO}$	100	100	241	321	280.7	2.39E + 01	186	228	208.6	1.27E + 01	214	250	231.6	1.33E + 01	63.205
ADAPTED-$RRT_{EX\text{-}GWO}$	100	100	**234**	322	278.5	2.58E + 01	**180**	216	203.4	1.26E + 01	**211**	235	222.9	8.85E + 00	**48.865**
GWO	100	100	239	340	283	3.34E + 01	191	250	217.1	1.94E + 01	233	281	258.2	1.81E + 01	153.763
I-GWO	100	100	241	328	276.7	2.95E + 01	204	251	224.2	1.68E + 01	219	284	253.4	2.57E + 01	161.690
EX-GWO	100	100	242	347	292.4	3.34E + 01	192	247	216.1	1.90E + 01	237	274	252.6	1.58E + 01	144.403
BPIB-RRT*	100	100	314	414	342.2	4.93E + 01	212	300	261.3	3.05E + 01	232	304	263.7	2.80E + 01	59.551
tGSRT	100	100	318	378	326.3	3.50E + 01	211	298	254.8	3.25E + 01	242	304	276.7	2.41E + 01	63.215
PSO	100	100	293	381	328.6	2.86E + 01	207	281	241.6	2.52E + 01	276	327	302	1.86E + 01	163.251
IBA	100	100	276	354	307.3	3.43E + 01	206	264	239.1	2.03E + 01	288	364	333.8	2.51E + 01	166.148
WOA	100	100	291	347	318.9	2.80E + 01	209	267	236.2	2.06E + 01	286	367	332.9	2.60E + 01	162.259

The bold values are the best results

Table 9 Simulation results for medium crowded map

Algorithm	Pop	Iter	UAV_1 Cost				UAV_2 Cost				UAV_3 Cost				Overall Time (s)
			Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
ADAPTED-RRT_{GWO}	25	40	286	389	336.8	3.67E + 01	216	267	238.9	1.50E + 01	240	307	207.7	2.17E + 01	**7.013**
ADAPTED-$RRT_{I\text{-}GWO}$	25	40	274	335	300.7	2.01E + 01	**198**	233	215.6	1.07E + 01	**237**	330	292.3	3.20E + 01	7.412
ADAPTED-$RRT_{EX\text{-}GWO}$	25	40	**270**	331	303.8	1.97E + 01	213	261	231.7	1.72E + 01	259	354	295.6	3.05E + 01	7.328
GWO	25	40	325	590	471	7.75E + 01	204	392	292.6	6.76E + 01	265	352	315.3	3.06E + 01	8.576
I-GWO	25	40	331	474	387.5	4.15E + 01	202	322	253.7	4.77E + 01	257	378	312.6	4.36E + 01	9.683
EX-GWO	25	40	327	451	398.5	4.02E + 01	199	343	276.7	5.57E + 01	255	388	312.1	5.06E + 01	9.756
BPIB-RRT*	25	40	290	425	361.8	4.19E + 01	224	401	323.2	6.94E + 01	316	425	364.4	3.64E + 01	8.478
tGSRT	25	40	334	454	393.3	3.43E + 01	230	400	328.2	7.30E + 01	309	433	368.3	4.09E + 01	9.367
PSO	25	40	340	466	403.7	4.60E + 01	263	461	348.5	6.64E + 01	354	466	405.1	4.40E + 01	9.962
IBA	25	40	397	489	411.4	3.71E + 01	274	462	377.2	6.83E + 01	312	478	407.8	6.13E + 01	10.241
WOA	25	40	398	484	444.9	3.17E + 01	249	412	321.6	5.74E + 01	312	476	415	5.72E + 01	9.985
ADAPTED-RRT_{GWO}	50	100	257	370	305.4	3.73E + 01	**188**	234	212.2	1.48E + 01	227	340	300.4	2.98E + 01	31.912
ADAPTED-$RRT_{I\text{-}GWO}$	50	100	240	305	278.6	2.44E + 01	205	243	222.8	1.45E + 01	**225**	305	278.6	2.44E + 01	40.332
ADAPTED-$RRT_{EX\text{-}GWO}$	50	100	**242**	323	287.2	2.58E + 01	190	223	209.3	1.01E + 01	226	323	287.2	2.58E + 01	**30.719**
GWO	50	100	322	432	367.2	4.13E + 01	192	278	240.3	2.92E + 01	261	432	367.2	4.13E + 01	54.028
I-GWO	50	100	317	429	376.6	3.49E + 01	197	303	251.1	3.73E + 01	237	429	367.6	3.49E + 01	41.292
EX-GWO	50	100	304	444	380.8	4.85E + 01	201	275	234.1	2.26E + 01	237	444	380.8	4.85E + 01	58.861
BPIB-RRT*	50	100	278	454	363.3	6.19E + 01	193	325	266.7	5.05E + 01	293	439	363.2	5.68E + 01	44.397
tGSRT	50	100	287	439	368.2	5.60E + 01	200	348	274.1	5.08E + 01	288	434	365.6	5.53E + 01	46.547
PSO	50	100	352	431	383	3.36E + 01	214	378	294.8	5.77E + 01	285	431	361.6	4.80E + 01	61.248
IBA	50	100	356	440	396.4	2.93E + 01	223	382	302.5	5.75E + 01	298	440	381.3	4.71E + 01	64.147
WOA	50	100	375	438	400.7	2.71E + 01	230	324	283.5	3.25E + 01	305	432	383.4	4.13E + 01	55.169
ADAPTED-RRT_{GWO}	100	100	247	324	288.9	2.99E + 01	**227**	312	268.6	2.79E + 01	218	259	238	1.39E + 01	71.314
ADAPTED-$RRT_{I\text{-}GWO}$	100	100	**230**	297	259.7	2.66E + 01	229	299	257.6	2.45E + 01	214	260	232.8	1.67E + 01	68.451
ADAPTED-$RRT_{EX\text{-}GWO}$	100	100	238	314	279.5	2.82E + 01	238	283	257.3	1.76E + 01	**211**	262	232.2	1.63E + 01	**61.568**
GWO	100	100	324	424	380.6	3.39E + 01	324	432	381.4	4.08E + 01	233	311	276.2	2.66E + 01	82.870
I-GWO	100	100	309	390	347	2.82E + 01	309	411	364	3.89E + 01	219	325	288.6	3.74E + 01	88.927
EX-GWO	100	100	303	395	352.6	3.19E + 01	300	386	343.6	3.44E + 01	237	332	286.5	4.33E + 01	94.854
BPIB-RRT*	100	100	259	386	326.2	4.50E + 01	281	454	358.5	5.62E + 01	232	377	319.9	4.79E + 01	73.147
tGSRT	100	100	279	385	324	3.91E + 01	298	459	370.5	4.92E + 01	242	360	314.3	4.07E + 01	77.578
PSO	100	100	357	448	409.8	3.07E + 01	324	509	382.3	6.14E + 01	276	385	327.2	4.08E + 01	97.263
IBA	100	100	344	450	403.7	3.52E + 01	310	481	370.4	5.84E + 01	288	401	344.8	4.01E + 01	99.589
WOA	100	100	345	437	398.1	3.04E + 01	327	460	387.5	4.92E + 01	286	396	344.1	3.94E + 01	96.328

The bold values are the best results

Table 10 Simulation results for large map

Algorithm	Pop	Iter	UAV_1 Cost				UAV_2 Cost				UAV_3 Cost				Overall Time (s)
			Best	Worst	Mean	Std	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
ADAPTED-RRT_{GWO}	25	40	379	501	443.9	4.05E + 01	**327**	440	377.1	3.57E + 01	**349**	435	394.2	2.71E + 01	**7.146**
ADAPTED-$RRT_{I\text{-}GWO}$	25	40	380	504	432.9	3.94E + 01	332	501	407.1	5.61E + 01	380	436	404.3	2.00E + 01	7.587
ADAPTED-$RRT_{EX\text{-}GWO}$	25	40	391	500	440.2	3.63E + 01	330	460	387.5	3.81E + 01	377	413	393	1.13E + 01	7.347
GWO	25	40	**363**	616	482	8.24E + 01	419	656	563.1	7.94E + 01	397	444	393.4	3.01E + 01	8.933
I-GWO	25	40	419	584	486	6.03E + 01	348	674	521.7	1.07E + 02	391	457	416.6	2.39E + 01	8.791
EX-GWO	25	40	**363**	558	470.8	6.60E + 01	464	685	579.1	7.45E + 01	388	472	431.4	3.14E + 01	8.136
BPIB-RRT*	25	40	400	625	528.8	8.59E + 01	340	870	596.8	1.58E + 02	404	625	529.2	8.53E + 01	8.896
tGSRT	25	40	410	653	550.8	7.84E + 01	350	850	630.2	1.51E + 02	397	653	549.5	8.10E + 01	8.678
PSO	25	40	400	626	529.7	8.05E + 01	501	774	615.3	8.98E + 01	432	639	534.2	7.71E + 01	9.874
IBA	25	40	415	616	532.5	6.38E + 01	487	790	638.5	1.03E + 02	416	616	532.6	6.36E + 01	10.548
WOA	25	40	391	587	477.8	6.13E + 01	410	720	582.7	1.02E + 02	431	587	481.8	5.61E + 01	9.866
ADAPTED-RRT_{GWO}	50	100	427	511	467.3	2.69E + 01	322	436	381.1	3.86E + 01	341	421	384.5	2.58E + 01	31.652
ADAPTED-$RRT_{I\text{-}GWO}$	50	100	384	501	432.1	3.50E + 01	334	498	408.2	4.85E + 01	**337**	424	367.9	2.97E + 01	**28.851**
ADAPTED-$RRT_{EX\text{-}GWO}$	50	100	387	471	437.5	2.57E + 01	**311**	426	366.3	3.23E + 01	344	415	382.4	2.39E + 01	30.948
GWO	50	100	374	528	444.8	5.32E + 01	340	624	470.4	9.36E + 01	362	461	402	3.33E + 01	51.601
I-GWO	50	100	383	512	452.3	4.89E + 01	344	625	484.4	9.64E + 01	345	467	399.8	3.66E + 01	56.879
EX-GWO	50	100	**358**	557	453.1	5.92E + 01	378	606	527.4	8.00E + 01	343	450	380.4	3.52E + 01	53.491
BPIB-RRT*	50	100	412	625	530	8.40E + 01	350	607	479.3	9.27E + 01	340	620	521.8	9.61E + 01	32.369
tGSRT	50	100	430	653	552.8	7.45E + 01	343	624	502.8	1.10E + 02	341	633	521.5	1.05E + 02	32.945
PSO	50	100	410	626	530.7	7.88E + 01	420	626	531.7	7.71E + 01	412	609	529.2	7.63E + 01	53.616
IBA	50	100	430	616	534	6.09E + 01	441	624	517	6.17E + 01	452	631	537.7	5.94E + 01	54.364
WOA	50	100	400	587	478.7	5.99E + 01	426	617	495	6.99E + 01	437	612	484.9	6.10E + 01	51.156
ADAPTED-RRT_{GWO}	100	100	396	458	434.4	1.87E + 01	312	419	371.4	3.66E + 01	338	432	394.6	3.23E + 01	70.464
ADAPTED-$RRT_{I\text{-}GWO}$	100	100	383	458	413.8	2.57E + 01	325	442	388.3	4.03E + 01	327	402	353.7	2.89E + 01	65.150
ADAPTED-$RRT_{EX\text{-}GWO}$	100	100	378	413	396	1.25E + 01	**305**	393	361.2	2.69E + 01	330	390	369.7	2.31E + 01	**65.072**
GWO	100	100	365	468	433.1	4.25E + 01	328	529	404	6.77E + 01	352	490	392.3	5.30E + 01	128.266
I-GWO	100	100	374	482	430.8	3.55E + 01	322	493	383.7	5.81E + 01	338	495	380.2	5.81E + 01	128.216
EX-GWO	100	100	**352**	547	448.6	6.52E + 01	339	483	390.8	4.60E + 01	346	471	380.8	4.56E + 01	118.059
BPIB-RRT*	100	100	400	620	527.9	8.48E + 01	374	626	519.7	9.03E + 01	**340**	620	521.9	9.62E + 01	72.579
tGSRT	100	100	413	623	548.1	7.39E + 01	381	629	547.5	8.24E + 01	349	624	522.2	1.01E + 02	74.364
PSO	100	100	400	603	527	7.77E + 01	385	609	530.5	7.78E + 01	364	606	524.1	8.51E + 01	136.362
IBA	100	100	431	611	533.6	6.00E + 01	346	618	494.8	9.73E + 01	347	613	511	9.37E + 01	130.268
WOA	100	100	425	596	479.6	6.18E + 01	371	603	479.4	6.81E + 01	373	599	477.2	6.67E + 01	129.365

The bold values are the best results

Table 11 Ranking summary of used algorithms in cost path parameter

Algorithm	Success rate (percent)	Rank
Adapted-RRT$_{GWO}$	11.11	5
Adapted-RRT$_{I\text{-}GWO}$	19.44	2
Adapted-RRT$_{EX\text{-}GWO}$	**27.78**	**1**
GWO	2.78	6
I-GWO	16.67	3
EX-GWO	13.88	4
BPIB-RRT*	2.78	6
tGSRT	0.00	10
PSO	2.78	6
IBA	0.00	10
WOA	2.78	6

The bold value and underlined value are the best results

experiments demonstrate that the metaheuristic algorithms need at least 40 iteration number and 25 population size for the optimal path, after which the metaheuristic algorithms are less likely to decrease the cost of the path. This issue has also been determined in studies in the literature [17, 22, 39].

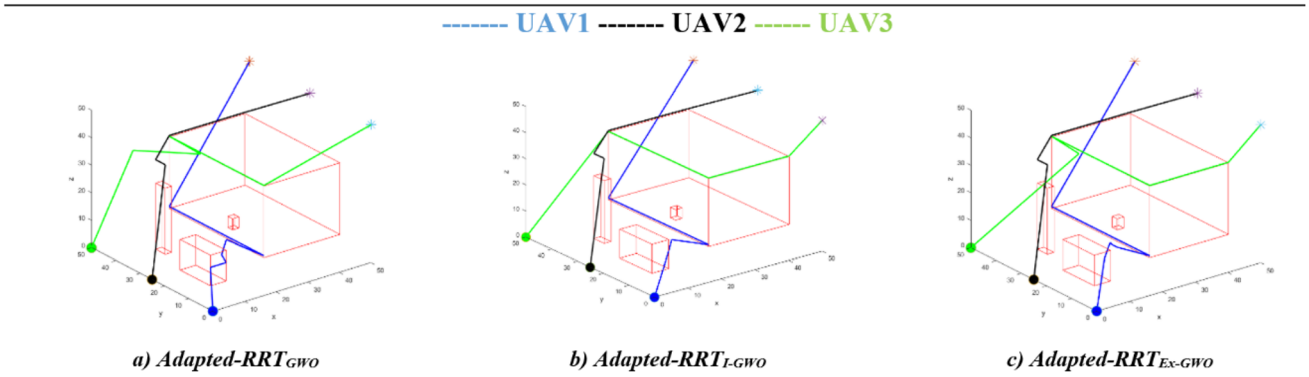
4.3 Comparison and evaluation of the execution time

The execution times of each algorithm are presented in Fig. 10. The proposed algorithms have better performance in terms of time taken when going through the maps. The execution time difference among the proposed algorithms and others are detailed in the figure. The performance of the execution time of Adapted-RRT$_{Ex\text{-}GWO}$ and Adapted-RRT$_{I\text{-}GWO}$ is better than others, respectively. In the Adapted-RRT method, locating the next station necessitates only the angle and the distance of the destination and the next station selection is based on the situation. Therefore, it can find suitable solutions in a short time. So, this is due to its hybrid mechanism. Another important inference is that as the population and iteration numbers of metaheuristic algorithms increase, the overall time parameter will also increase, but this problem is not encountered despite using metaheuristic algorithms in the proposed hybrid versions.

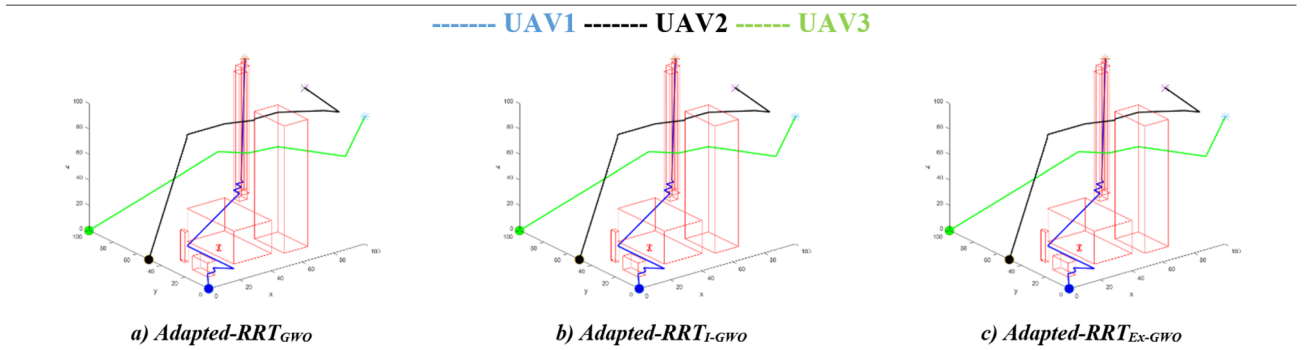
Also, in time complexity analysis, while metaheuristic-based methods generally have $O(n^2) \leq T$ notations, sampling-based methods such as RRT have $O\ (n\ \text{Log}n) \leq T \leq O(n^2)$ complexity. Our proposed methods, which are a hybridization of sampling and metaheuristic methods, have $O(n^2)$ complexity. In the computational analysis, the Adapted-RRT$_{I\text{-}GWO}$ is better than the two other proposed. Because I-GWO depends on only alpha wolf and therefore, calculations are more efficient. Among the other two algorithms (Adapted-RRT$_{GWO}$ and Adapted-RRT$_{Ex\text{-}GWO}$), the best performance belongs to Adapted-RRT$_{GWO}$. Because only the best three wolves are taken into account in Adapted-RRT$_{GWO}$. However, in Adapted-RRT$_{Ex\text{-}GWO}$, every wolf in the group follows all their previous wolves and performs an update process. On the other hand, the complexity of BPIB-RRT * is $O(n\ \text{Log}n)$, tGSRT is $O(n^2)$, and the complexity of the rest of the other algorithms is equal or bigger than $O(n^2)$.

4.4 Comparison and evaluation of the convergence rate

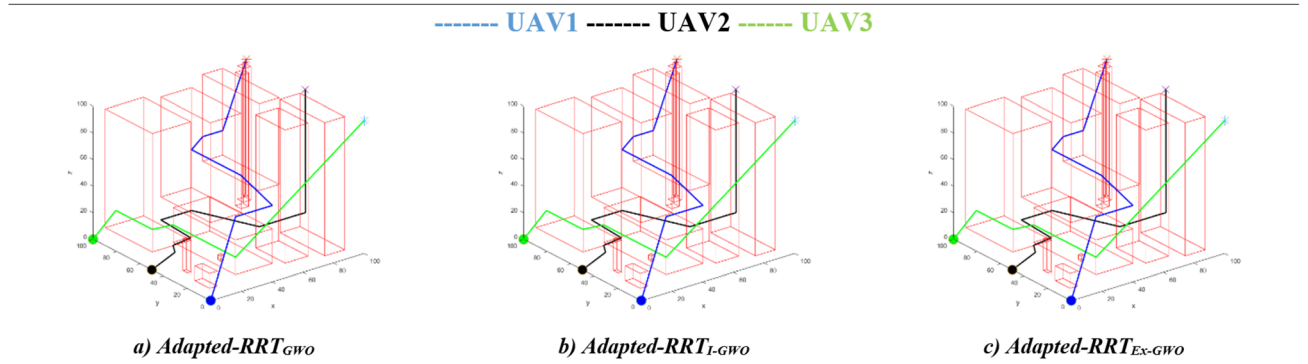
Another analysis parameter is the convergence rate. Figures 11, 12, and 13 present the convergence curve of each path planning algorithm. As aforementioned, the obstacles numbers and the boundary sizes of the map are declared in Sect. 3.2. The three versions of the proposed path planning method have different structures with respect to exploration and exploitation. The presented figures illustrate the convergence curve of algorithms with 40 iterations and a population size of 25. In general metaheuristic algorithms have better convergence performances than others. Among the metaheuristic approaches used in this study, GWO-based methods are more successful due to their working mechanism as a swarm-intelligence group. Because wolves can update their positions according to the positions of other wolves in the herd and approach the hunt (solution). Based on the results, the Ex-GWO-based Adapted-RRT algorithm has better convergence curve behaviour overall in many cases, because in the Ex-GWO algorithm the experience of the entire population is used in the optimal pathfinding, in each iteration. When the convergence curve behaviours are analysed, it is seen that the Adapted-RRT$_{Ex\text{-}GWO}$ method generally behaves more balanced between exploration and exploitation phases. With this balance, it is slowly moving towards a global optimum. Thus it gives a chance to find all possible candidate solutions in the global workspace. The term convergence in the literature [20, 55, 57–59] refers to the rate or behaviour of an algorithm towards the global optimum. Of course, a quick convergence results in local optima stagnation. By contrast, sudden changes in the solutions lead to local optima avoidance but reduce the convergence speed towards the global optimum. In other words, the search agents change abruptly in the early stages of the optimization process and



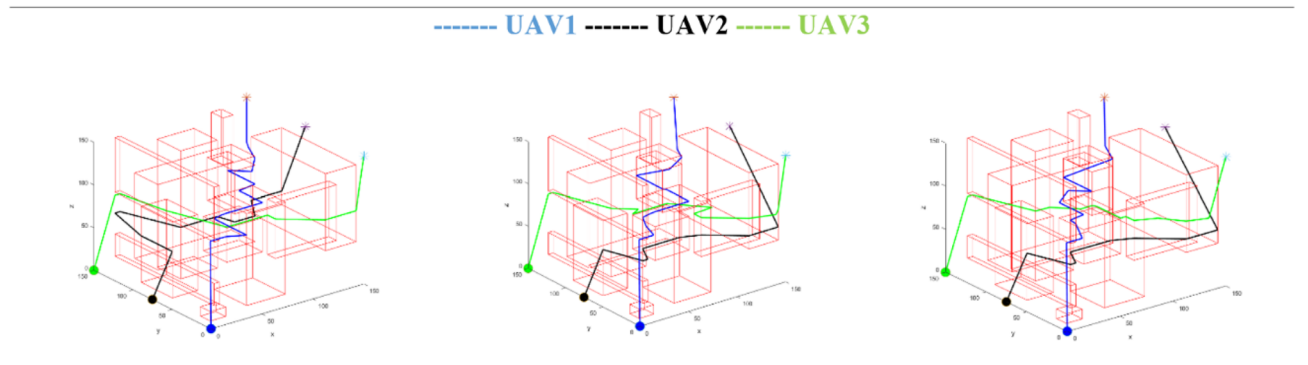
(a) Generated optimal paths for small map population:25 and iteration:40



(b) Generated optimal paths for medium map population:25 and iteration:40

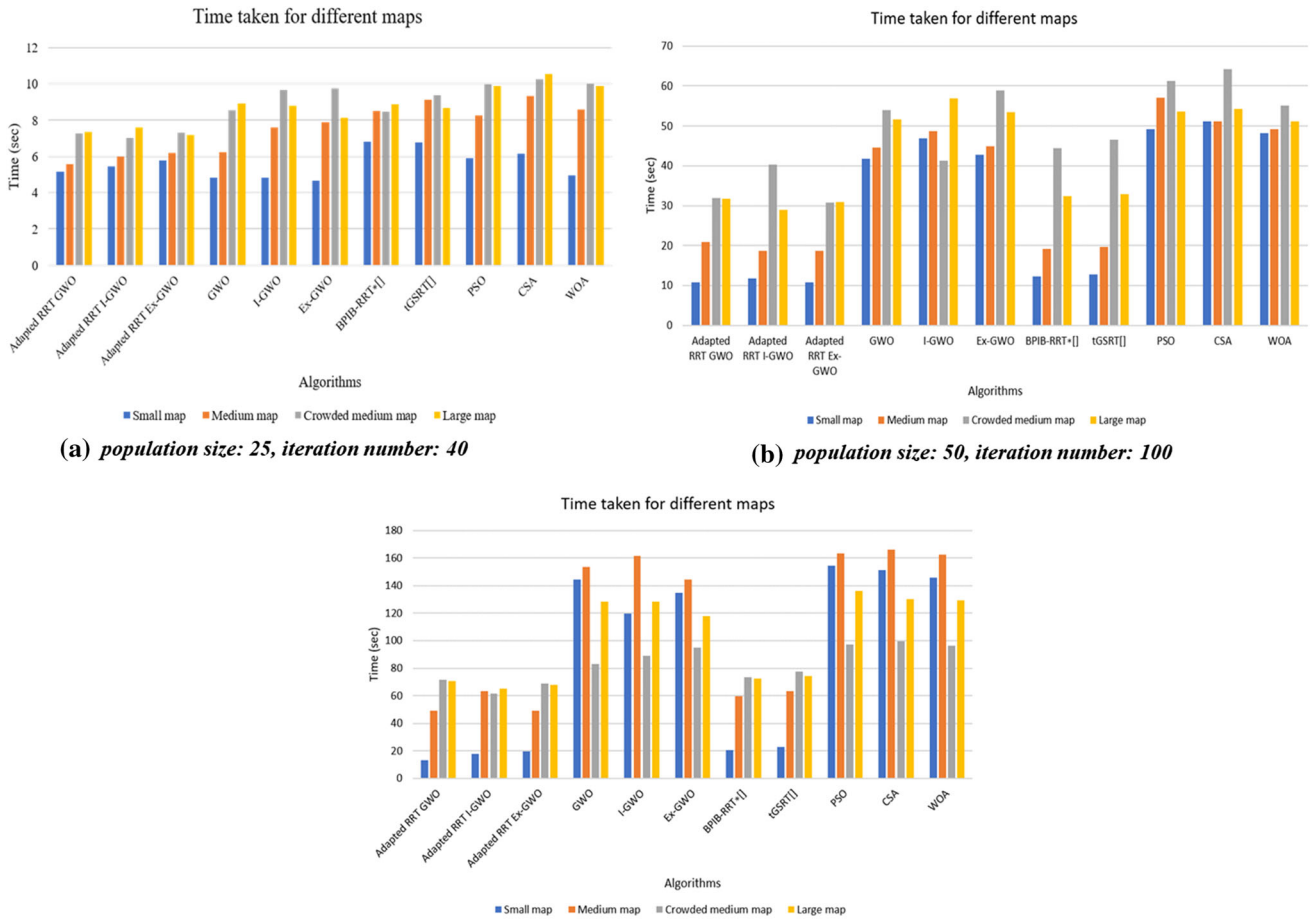


(c) Generated optimal paths for crowded medium map population:25 and iteration:40



(d) Generated optimal paths for large map population:25 and iteration:40

Fig. 9 Generated optimal paths in four maps



(a) ***population size: 25, iteration number: 40***

(b) ***population size: 50, iteration number: 100***

(c) ***population size: 100, iteration number: 100***

Fig. 10 Execution time analysis for different maps in each algorithm

then gradually converge. The proposed method generally has a balanced movement in all situations where it could find or not the best solution. In other words, using the Ex-GWO algorithm, this method mainly exhibits balanced convergence behaviour. Also, thanks to its characteristic feature, it outperformed in more complex and larger areas. According to the results and the figures obtained, regarding the I-GWO algorithm, UAVs reach the best optimal path earlier than when using other techniques. In the analysis, different iteration sizes have been considered to get the size of the optimal iteration. As a result of our observations, we conclude that 40 iterations are enough to find the best path. Because it was seen that the same results were obtained when more iterations were continued. The acquired outcomes also indicate that the execution time of algorithms is variable in the maps. Due to the number of obstacles and the map boundary, different results are collected. Also, while using three UAVs with different initial and final states' behaviours in the convergence rate analysis, it is shown that the number of obstacles has an effect on the path cost.

Figure 14 depicts the boxplot analysis of each map and UAV_1. The aim is to compare the performance of the proposed and used methods. These results are acquired for 10 runs of metaheuristics with iteration size of 40. The box plot graph analysis shows the maximum and minimum values of the best cost obtained along with the frequency of the values. The boxplot analysis shows that Adapted-$RRT_{Ex\text{-}GWO}$ algorithm gives better performance in comparison with other algorithms. It can be stated, based on values in Sect. 4.2, the finding costs by Ex-GWO-based methods are generally close to the average values, shown by red lines in the Fig. 14. It should be noted that best cost was typically obtained after an average of 10 runs.

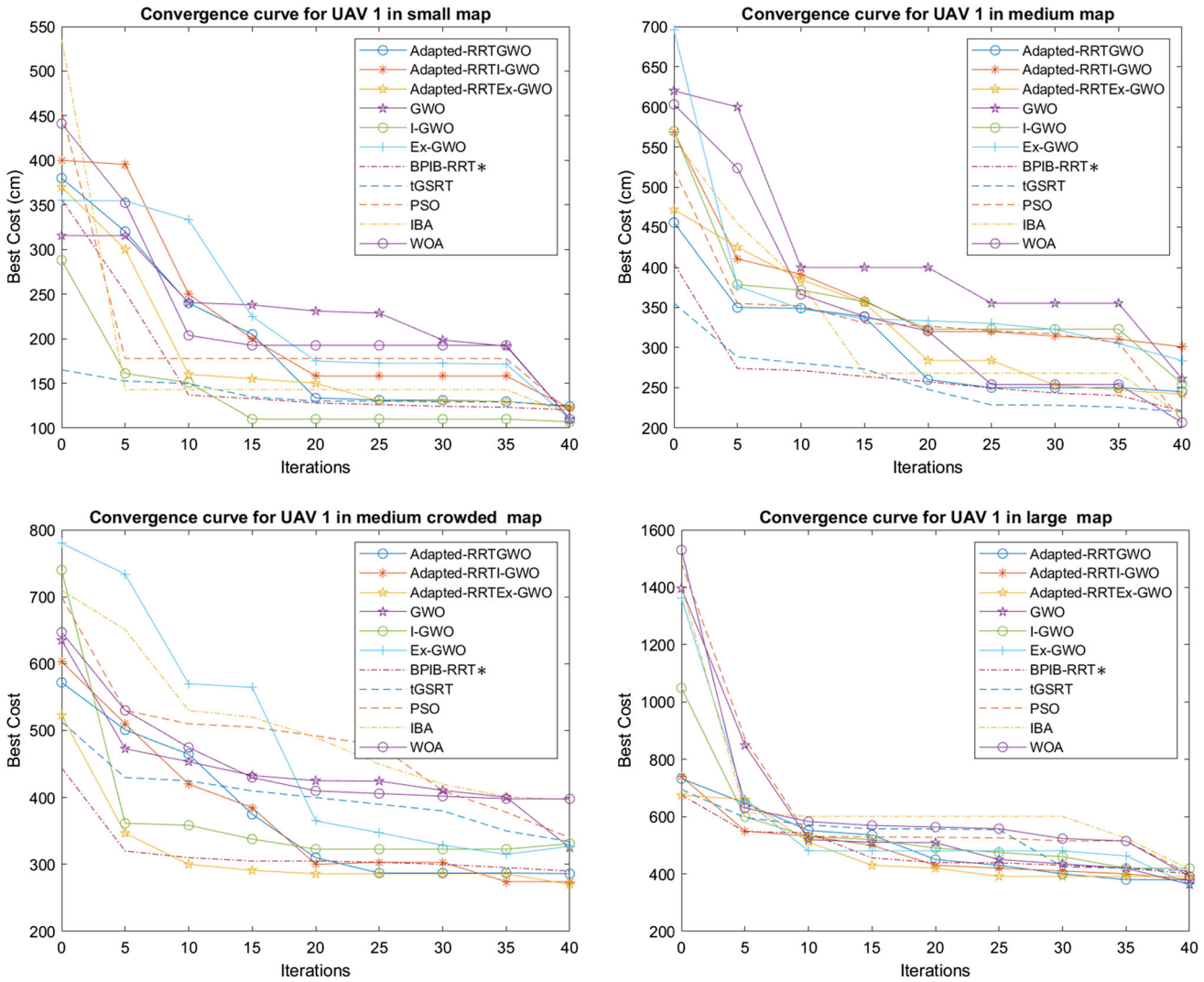


Fig. 11 Convergence analysis for UAV_1 on each map with 25 populations and 40 iterations

4.5 General comparison and discussion

Table 12 presents the general characteristics of each type of method in path planning. In this paper, we compared our proposed three-versions method with 8 different methods (sampling and nature categories-based). The proposed method has been evaluated in a simulation environment like other algorithms used to compare. In this simulation, the obtained results and analysis of them explain the proposed method outperformed other algorithms in defined considered parameters. As a result, a summary evaluation of our proposed hybrid method with other methods in both categories has also been made. The values and ranges of the parameters in this study have been used considering the parameters and their ranges as they appear in the literature [18, 23, 27, 29, 43, 56].

The advantages and outperform of the method proposed in this study are discussed in the simulation and analysis section. However, this method did not take into account UAVs' own variables such as the battery, speed, altitude, and processing power. In addition, dynamic conditions such as wind and rain in real environments are not taken as a basis. However, the proposed method is also expected to be successful under these conditions, in line with its general architecture and hybrid mechanism. Since this method has a wide range, it can be used in real and various application areas, and new studies are planned in the future based on the parameters that are not considered in this study.

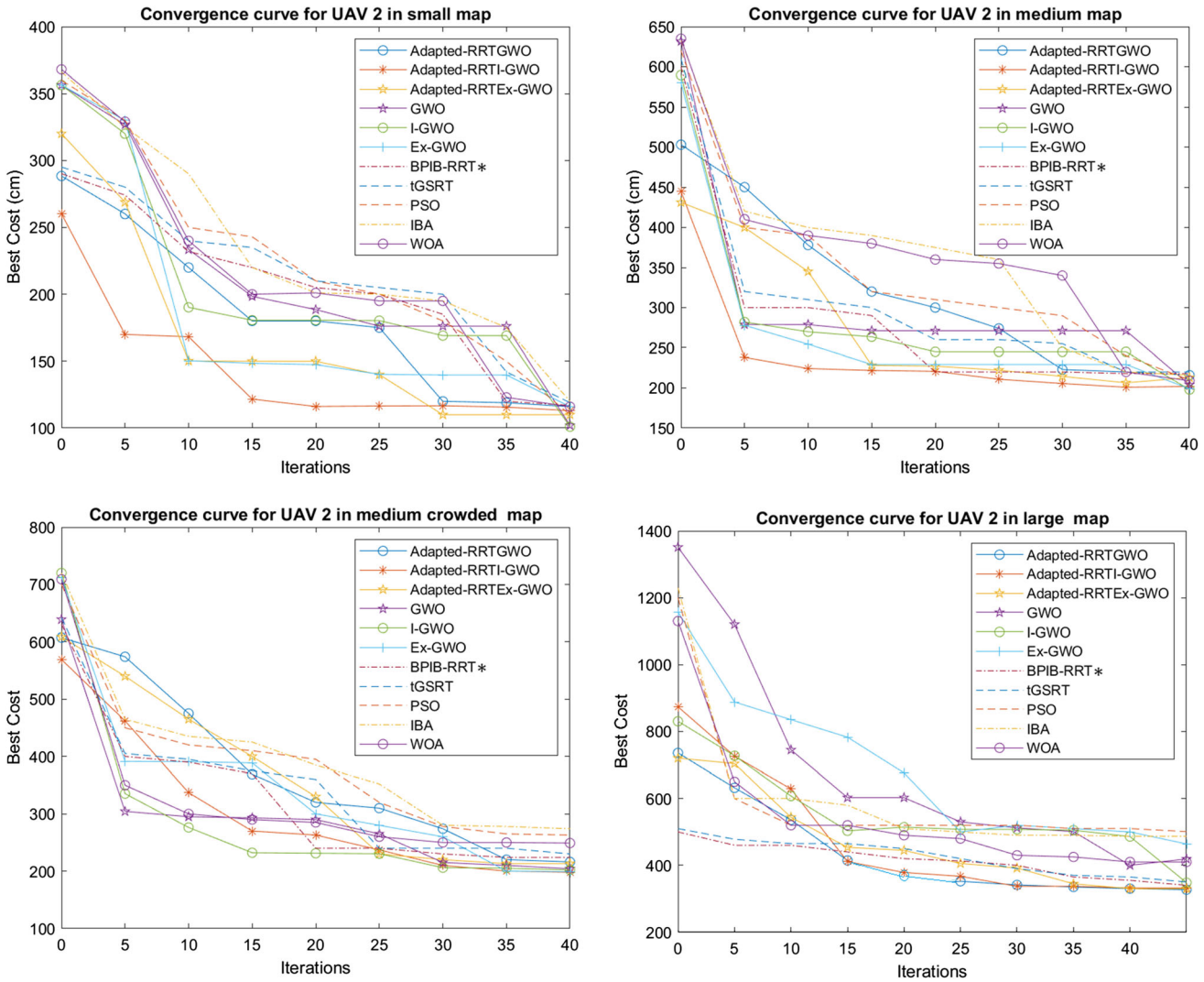


Fig. 12 Convergence analysis for UAV_2 on each map in 25 populations and 40 iterations

5 Conclusions and future scope

This study presented a new hybrid method with three versions to solve 3D path planning problem in autonomous UAVs. In this paper, an improvement on RRT algorithm, namely Adapted-RRT, is presented and it is applied with three different well-known metaheuristic algorithms; GWO, I-GWO, and Ex-GWO. These methods are named as Adapted-RRT_{GWO}, Adapted-$RRT_{I\text{-}GWO}$, and Adapted-$RRT_{Ex\text{-}GWO}$. They were suggested to find the optimal paths between initial and final stations of each UAV without any collision as well as with minimum time and space complexities. Adapted-RRT avoids possible obstacles without making random movements. In the Adapted-RRT were deployed two important operations (length of each movement and direction of each movement) to select next stations and finally generate path. For this, three metaheuristic algorithms are used to perform these two operations in the most appropriate way. Based on results, it was concluded that the Adapted-$RRT_{EX\text{-}GWO}$ can perform more successfully in larger and crowded environments with more obstacles, populations and iterations. On the other hand,

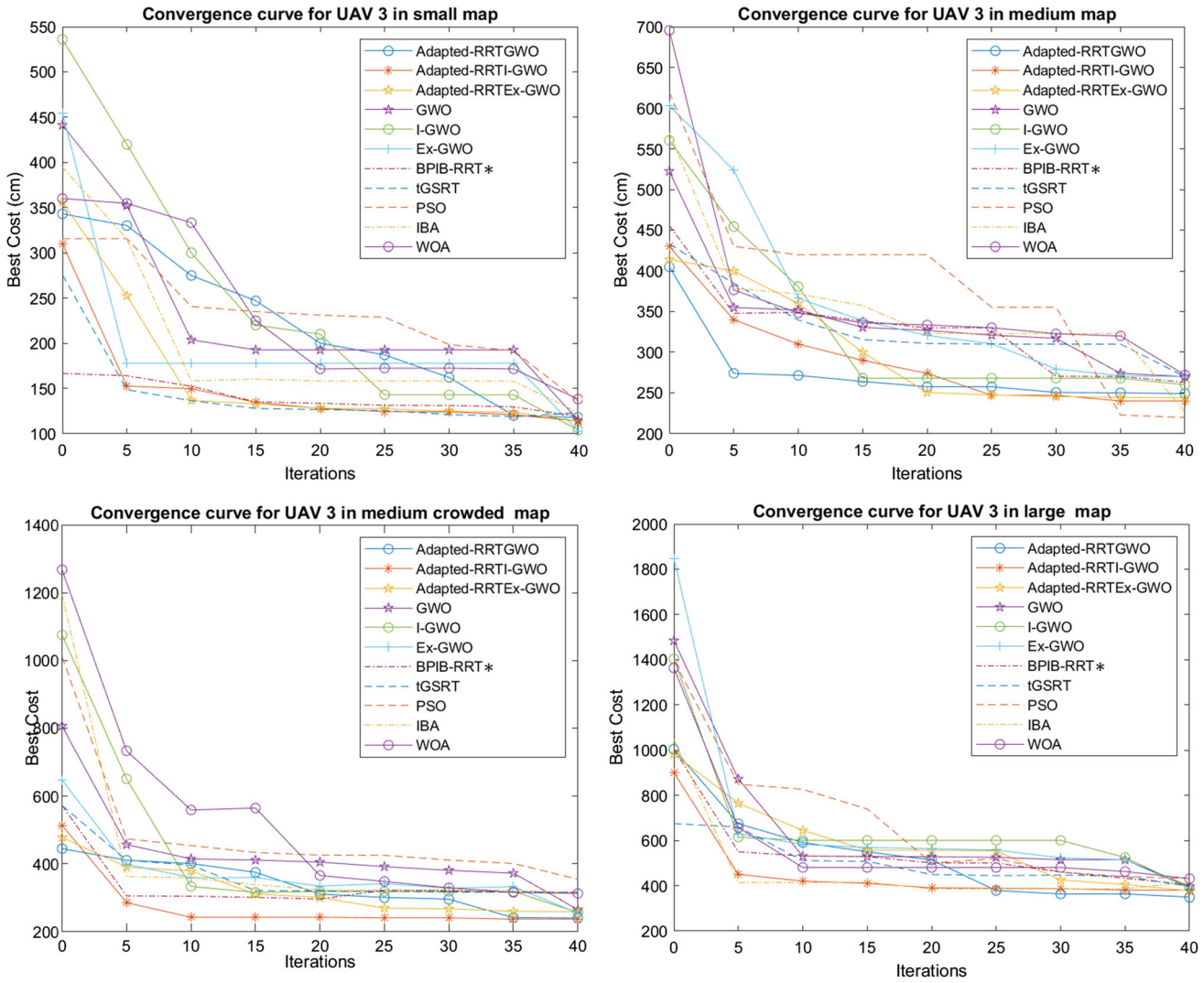


Fig. 13 Convergence analysis for UAV_3 on each map with 25 populations and 40 iterations

Adapted-$RRT_{I\text{-}GWO}$ method may give good results in small-medium sized environments. In addition, it may be more successful than other GWO algorithms in fewer populations and iterations. The GWO-based method is generally between these two methods and behaves in a balanced way. In this study, four different maps with various obstacles have been used, furthermore, three UAVs with different start and destination states have been considered. The proposed methods have been analysed in terms of optimal path costs, execution time, and convergence rate by varying the population sizes as well as the iteration numbers. The results obtained show that the Adapted-$RRT_{Ex\text{-}GWO}$ method outperforms the others 10, out of the total of 11 algorithms. In future work, 3D path planning method proposed will be realized using machine learning methods such as reinforcement learning or game theory-based algorithms. Another possible direction of further study can be path planning for Autonomous Underwater Vehicle (AUV), connected vehicles with VANET and FANET structures.

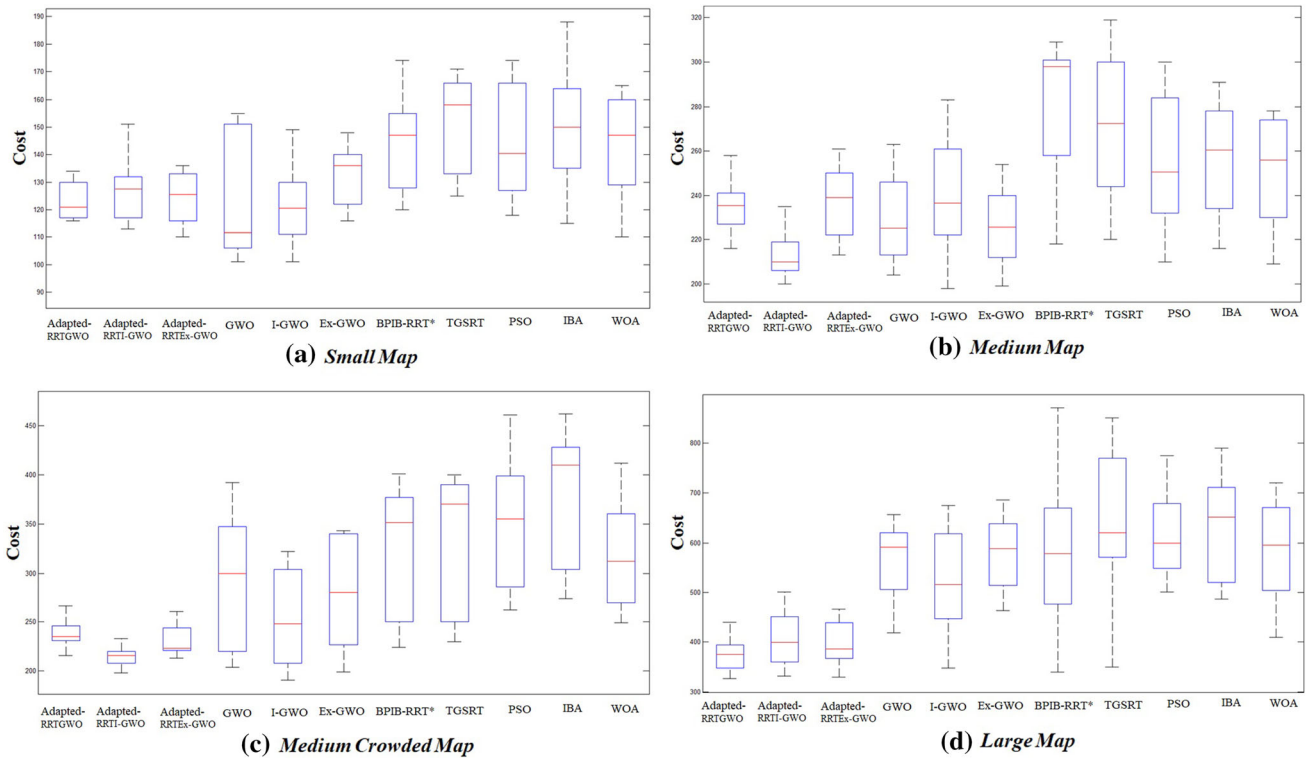


Fig. 14 Boxplots graph analysis for UAV_2 for each map with 25 populations and 40 iterations

Table 12 Features of the focused path planning methods

Features	Sampling-based algorithms	Nature-based algorithms	Our method
Obstacle avoidance	Yes	Yes	Yes
Optimum pathfinding	No	Yes	Yes
Environment	Small	Complex	Complex
Convergence	Slow	Fast	Fast
Local minima capture	Yes	No	No
Operating environment	Static/dynamic	Static/dynamic	Static/dynamic
Time efficiency in find a path	High	Low	Medium
Time efficiency in optimal pathfinding	Low	Medium	High
Searching	Fast	Medium	Medium
Process costs	Low	Medium	Low
Computational overload	High	Medium	Medium
Complexity	$O(n\ \mathrm{Log}n) \leq T \leq O(n^2)$	$O(n^2) \leq T$	$O(n^2)$
Device memory efficiency	Medium–high	Low	Medium–high
Multi-paths finding	No	Yes	Yes
Random movement	Yes	Partial	No
Location and number of intermediate stations	Undetermined	Predetermined	Undetermined
Intermediate stations selection	Usually random but a certain mechanism is followed	Optimized selection but on a limited candidate stations	Optimized selection from desired number of candidate stations
Adaptability	High	Low	High
Fault tolerance	Low	High	High

Appendix: Terminologies

Term	Explanation
Path Planning	It is a computational problem to find a sequence of valid configurations that moves the mobile robot from the source to the destination
Optimal Path Planning	It is measured based on various factors such as path length, collision-free space, execution time, and the total number of turns
Sampling-based Path Planning	These methods need some pre-known information of the whole workspace, that is, a mathematical representation to describe the workspace
Mathematical-based Path Planning	These algorithms model the environment (kinematic constraints) as well as the parameters of the system (dynamic). Afterward, they map the bounds of these two parameters based on the cost function bound
Nature-based Path Planning	These methods attempt to find an almost optimal path by eliminating the process of creating complex environment models based on stochastic approaches
Node-based Path Planning	These algorithms are informed search methods and find an optimal path based on certain decomposition
Unknown Environments	The mobile robot does not have any information about the environment. In this case, the movements are not deterministic and it is according to local information. The results for all actions are unknown to the agent
Known Environments	The mobile robot has enough information about the environment. the results for all actions are known to the agent
Fully or Partially Known Environments	The whole or part of the environment refers to certain situations
Unstructured Environments	The mobile robot cannot rely on complete knowledge about its environment. The environment fills with uncertain elements
Trajectory	The path between two points that followed by an object
Metaheuristics Algorithms	These algorithms try to efficiently explore the search space in order to find near-optimal solutions by exploiting without falling to local trap optima
Local Optima Trap	Situation mistakenly considered the best solution. However, there is a better solution in the global area than the found solution
Autonomous Mobile Robots	They gain required information about their environment. They can operate autonomously without human intervention

Acknowledgements The open-source code of the proposed path planning method is presented at the https://github.com/aliyevroyal/3DPathPlanningForUAVs.

Declarations

Conflict of interest The author declares that they have no conflict of interest.

References

1. Kiani F, Nematzadehmiandoab S, Seyyedabbasi A (2019) Designing a dynamic protocol for real-time Industrial Internet of things-based applications by efficient management of system resources. Adv Mech Eng 11(10):1–23
2. Ha QP, Yen L, Balaguer C (2019) Robotic autonomous systems for earthmoving in military applications. Autom Constr 107(102934):1–19
3. Kiani F (2017) Reinforcement learning based routing protocol for wireless body sensor networks. In: IEEE 7th international symposium on cloud and service computing (SC2), pp 71–78
4. Sumi L, Ranga V (2018) An IoT-VANET-based traffic management system for emergency vehicles in a smart city. Recent Findings Intell Comput Tech 708:23–31
5. Bacco M et al (2018) Reliable M2M/IoT data delivery from FANETs via satellite. Int J Satell Commun Netw 37(4):1–12
6. Nayyar A, Nguyen BL, Nguyen NG (2020) The internet of drone things (IoDT): future envision of smart drones. In: First international conference on sustainable technologies for computational intelligence. Advances in intelligent systems and computing. Springer, vol 1045, pp 563–580
7. Chen Y, Lu C, Chu W (2020) A cooperative driving strategy based on velocity prediction for connected vehicles with robust path-following control. IEEE Internet Things J 7(5):3822–3832
8. Nayyar A, Le DN, Nguyen NG (eds) (2018) Advances in swarm intelligence for optimizing problems in computer science. CRC Press, Boca Raton
9. Nayyar A, Nguyen NG (2018) Introduction to swarm intelligence. Advances in swarm intelligence for optimizing problems in computer science, pp 53–78
10. Wolpert DH, Macready WG et al (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
11. Flemming S, la Anders CH, Morten B (2011) Configuration space and visibility graph generation from geometric workspaces for UAVs, book section 4. American Institute of Aeronautics and Astronautics, Reston
12. Bera T, Bhat MS, Ghose D (2014) Analysis of obstacle based probabilistic roadmap method using geometric probability. IFAC Proc Elsevier 47(1):462–469
13. LaValle S (1998) Rapidly-exploring random trees: a new tool for path planning, Technical report, Computer Science Department, Iowa State University, Ames, Iowa, USA, pp 1–4
14. Kavraki LE, Svestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high dimensional configuration spaces. IEEE Trans Robot Autom 12(4):566–580
15. Oz I, Topcuoglu HR, Ermis M (2013) A Metaheuristic based three-dimensional path planning environment for unmanned aerial vehicles. Simulation Trans Soc Model Simul Int 89(8):903–920
16. Pandey P, Shukla A, Tiwari R (2018) Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm. Int J Syst Assur Eng Manag 9:836–852

17. Qu G, Gai W, Zhong M, Zhang J (2018) A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. Appl Soft Comput J 89(2020):1–12
18. Yang L, Qi J, Song D, Xiao J, Han J, Xia Y (2016) Survey of robot 3D path planning algorithms. J Control Sci Eng 2016:1–22
19. Noreen I, Khan A, Habib Z (2016) A comparison of RRT, RRT* and RRT*-smart path planning algorithms. Int J Comput Sci Netw Secur 16(10):20–27
20. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
21. Seyyedabbasi A, Kiani F (2021) I-GWO and Ex-GWO: improved algorithms of the grey wolf optimizer to solve global optimization problems. Eng Comput 37:509–532
22. Patle BK, Pandey A, Jagadeesh A, Parhid DR (2018) Path planning in uncertain environment by using firefly algorithm. Defence Technol 18(6):691–701
23. Patle BK, Babu LG, Pandey A, Parhi D, Jagadeesh A (2019) A review: On path planning strategies for navigation of mobile robot. Defence Technol 15(4):582–606
24. Maw AA, Tyan M, Lee JW (2020) iADA*: improved anytime path planning and replanning algorithm for autonomous vehicle. J Intell Rob Syst 100:1005–1013
25. Nayyar A, Nguyen NG, Kumari R, Kumar S (2020) Robot path planning using modified artificial bee colony algorithm. In: Frontiers in intelligent computing: theory and applications. Springer, Singapore, pp 25–36
26. Youn W, Ko H, Choi H et al (2020) Collision-free autonomous navigation of a small UAV using low-cost sensors in GPS-denied environments. Int J Control Autom Syst 18:1–16
27. Memmah MM, Lescourret F, Yao X et al (2015) Metaheuristics for agricultural land use optimization. A review. Agron Sustain Dev 35:975–998
28. Sanchez JL, Wang M, Olivares-Mendez MA et al (2019) A real-time 3D path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments. J Intell Rob Syst 93:33–53
29. Wang L, Kan J, Guo J, Wang C (2019) 3D path planning for the ground robot with improved ant colony optimization. Sensors 19(815):1–21
30. Yang L, Qi J, Xiao J, Yong X (2014) A literature review of UAV 3D path planning. In: Proceeding of the 11th world congress on intelligent control and automation, Shenyang, pp 2376–2381
31. Noreen I, Khan A, Habib Z (2018) Optimal path planning using RRT*-adjustable bounds. Intell Serv Robot 11(1):41–52
32. Lin Y, Saripalli S (2017) Sampling-based path planning for UAV collision avoidance. IEEE Trans Intell Transp Syst 18(11):3179–3192
33. Nash A, Koenig S, Tovey C (2010) Lazy theta*: any-angle path planning and path length analysis in 3D. In: Proceedings of the third annual symposium on combinatorial search, vol 2, pp 153–154
34. Guruji KA, Agarwal H, Parsediya DK (2016) Time-efficient A* algorithm for robot path planning. Proc Technol 23:144–149
35. Zhu Q, Yan Y, Xing Z (2006) Robot path planning based on artificial potential field approach with simulated annealing. In: Sixth international conference on intelligent systems design and applications, Jinan, pp 622–627
36. Tisdale T, Kim ZW, Hedrick JK (2009) Autonomous UAV path planning and estimation: an online path planning framework for cooperative search and localization. IEEE Robot Autom Mag 16(2):35–42
37. Ma CS, Miller RH (2006). Milp optimal path planning for real-time applications. In: Proceedings of the American control conference, pp 1–6
38. Jason G, Xin M, Liu F, Ying W, Ren H (2017) Mathematical modeling and intelligent algorithm for multi-robot path planning, Mathematical Problems in Engineering, pp 1–2
39. Choudhury N, Mandal R, Kar SK (2016) Bioinspired robot path planning using PointBug algorithm. In: 2016 international conference on electrical, electronics, and optimization techniques (ICEEOT), Chennai, pp 2638–2643
40. Dewangan RK, Shukla A, Godfrey WW (2019) Three dimensional path planning using Grey wolf optimizer for UAVs. Appl Intell 49(6):2201–2217
41. Abhishek B, Ranjit S, Shankar T et al (2020) Hybrid PSO-HSA and PSO-GA algorithm for 3D path planning in autonomous UAVs. SN Appl Sci 2(1805):1–16
42. Huang Y, Fei M (2018) Motion planning of robot manipulator based on improved NSGA-II. Int J Control Autom Syst 16:1878–1886
43. Panda M, Das B, Subudhi B et al (2020) A comprehensive review of path planning algorithms for autonomous underwater vehicles. Int J Autom Comput 17:321–352
44. Karaman S, Frazzoli E (2010) Optimal kinodynamic motion planning using incremental sampling-based methods. In: 49th IEEE conference on decision and control (CDC), pp 7681–7687
45. Chao N, Liu YK, Xia H, Ayodeji A, Bai L (2018) Grid-based RRT* for minimum dose walking path-planning in complex radioactive environments. Ann Nucl Energy 115:73–82
46. Chao N, Liu YK, Xia H, Peng MJ, Ayodeji A (2019) DLRRT* algorithm for least dose path re-planning in dynamic radioactive environments. Nucl Eng Technol 51(3):825–836
47. Jordan M, Perez A (2013) Optimal bidirectional rapidly-exploring random trees, technical report MIT-CSAIL-TR-2013-021, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 2013
48. Hidalgo-Paniagua A, Bandera JP, Ruiz-De-Quintanilla M, Bandera A (2018) Quad-RRT: a real-time GPU-based global path planner in large-scale real environments. Expert Syst Appl 99(1):141–154
49. Wu X, Xu L, Zhen R, Wu X (2019) Biased sampling potentially guided intelligent bidirectional RRT* algorithm for UAV path planning in 3D environment. Mathematical Problems in Engineering, 2019
50. Wang H, Wentao L, Peng Y, Xiao L, Chang L (2015) Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system. Chin J Aeronaut 28(1):229–239
51. Perazzo P, Sorbelli FB, Conti M, Dini G, Pinotti CM (2016) Drone path planning for secure positioning and secure position verification. IEEE Trans Mob Comput 16(9):2478–2493
52. Wang GG, Chu HE, Mirjalili S (2016) Three-dimensional path planning for UCAV using an improved bat algorithm. Aerosp Sci Technol 49:231–238
53. Cao X, Zou X, Jia C, Chen M, Zeng Z (2019) RRT-based path planning for an intelligent litchi-picking manipulator. Comput Electron Agric 156:105–118
54. Mirshamsi A, Godio S, Nobakhti A, Primatesta S, Dovis F, Guglieri G (2020) A 3D path planning algorithm based on PSO for autonomous UAVs navigation. Bioinspired optimization methods and their applications. BIOMA 2020, 12438, pp 268–280
55. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
56. Santos LC, Santos FN, Solteiro Pires EJ, Valente A, Costa P, Magalhães S (2020) Path Planning for ground robots in agriculture: a short review. In: 2020 IEEE international conference on autonomous robot systems and competitions (ICARSC), Ponta Delgada, Portugal, pp 61–66

57. Van den Bergh F, Engelbrecht AP (2006) A study of particle swarm optimization particle trajectories. Inf Sci 176(8):937–971
58. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
59. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191