SN Computer Science (2024) 5:286
https://doi.org/10.1007/s42979-024-02605-x

ORIGINAL RESEARCH

Efficient 3D Path Planning for Drone Swarm Using Improved Sine Cosine Algorithm

Probhat Pachung[1] · Kopal Pandya[2] · Atulya Nagar[3] · Jagdish Chand Bansal[1]

Received: 29 September 2023 / Accepted: 3 January 2024

Abstract
Path planning is one of the most important steps in the navigation and control of swarm of drones. It is primarily concerned with avoiding collision among drones and environmental obstacles while determining the most efficient flight path to the region of interest. Whenever there is a high density and complex mission, path planning becomes the most challenging and indispensable task. The problem of path planning is not only relevant to finding the optimum path from the start point to the destination point but also to provide a mechanism for preventing collisions on the path. Hence, an appropriate algorithm is needed to plan the optimal path for the swarm of drones. This paper proposes an efficient methodology for drone swarm path planning problems in 3D environments. An improved population-based meta-heuristic algorithm, Sine Cosine Algorithm (SCA), has been proposed to solve this problem. As part of the improvements, the population of SCA is initialized using a chaotic map, and a non-linearly decreasing step size is used to balance the local and global search. In addition, a convergence factor is employed to increase the convergence rate of the original SCA. The performance of the proposed improved SCA (iSCA) is tested over the drone swarm path planning problem, and the results are compared with those of the original SCA, and other state-of-the-art meta-heuristic algorithms. The experimental results show that the drone swarm 3D path planning problem can be efficiently handled with the proposed improved SCA.

Keywords Path planning · Internet of drones (IoDs) · Meta-heuristics · Sine cosine algorithm (SCA) · Drone swarm · Obstacle avoidance

Introduction

In recent years, the field of Unmanned Aerial Vehicles (UAVs) has grown rapidly, including miniature aircraft, airships, and drones for a wide range of purposes such as surveillance, military operations, telecommunications, medical supplies delivery, rescue operations, and monitoring [1–3]. A large number of UAV systems rely on only one aerial vehicle. Nevertheless, the active cooperation of several UAVs is essential in many applications. In addition to being cost-effective and more robust, they can perform complex tasks beyond the capacity of a single UAV, and more robust.

An Internet of Drones (IoDs) or drone swarm is a network of drones connecting to each other, a layered network control architecture that is primarily responsible for coordinating the access of UAVs, controlling airspace, and providing navigation services between nodes [4]. Drone swarm can be utilized in a variety of applications, including intelligent transportation systems (ITS) for improving vehicle-infrastructure communication. In this application, drone swarm

This article is part of the topical collection "Emerging Applications of Data Science for Real-World Problems" guest edited by Satyasai Jagannath Nanda, Rajendra Prasad Yadav and Mukesh Saraswat.

✉ Jagdish Chand Bansal
jcbansal@sau.ac.in

Probhat Pachung
probhatpachung23@gmail.com

Kopal Pandya
kopal.pandya@student.manchester.ac.uk

Atulya Nagar
atulya.nagar@hope.ac.uk

[1] Department of Mathematics, South Asian University, New Delhi 110068, New Delhi, India

[2] Department of Mechanical, Aerospace and Civil Engineering, University of Manchester, Oxford Road, Manchester M13 9PL, UK

[3] Department of Mathematics, Liverpool Hope University, Hope Park, Liverpool L16 9JD, UK

Published online: 24 February 2024

is an efficient way to improve traffic rules on the ground and provide ground users with efficient information dissemination. To accomplish such complex tasks, drones must collaborate due to the heterogeneity of their goals and communication technologies. In the current scenario, drones are becoming increasingly autonomous as technology advances, and they gain new capabilities. However, as drones get closer to each other or obstacles in case of high drone density or challenging missions, they pose new threats.

Obstacles can be static or dynamic. The static obstacles are fixed, such as mountains and buildings, while the dynamic obstacles include other drones or air vehicles, birds, etc. Furthermore, controlling drone swarm and communicating among drones become more complicated tasks. Moreover, if the drone swarm merges in different directions, a catastrophic collision is more likely to occur. Since the likelihood of collision among drones in a swarm increases, preventing or avoiding collisions becomes more challenging, and hence, drone swarm should have a proper collision-avoiding method.

One of the most important problems for autonomous multi-UAV system i.e. drone swarm is path planning. Considering the given flight conditions and flight environment, a collision-free path for drone swarm needs to be planned based on the given starting and destination points. The planned path should be cost-effective and comply with relevant constraints. Thus the drone swarm path planning can be viewed as an optimization problem that involves multiple constraints [5], and the objective is to find the shortest feasible path between one point and another point based on various optimization criteria and mission constraints [6, 7]. These constraints include the minimum flight length, minimum flight time, and state constraints of the drones. Recently, research on drone swarm path planning has received much attention since it enables unmanned systems to operate autonomously and intelligently.

In recent years, for UAVs and autonomous robots, several path planning algorithms have been proposed. In addition to Graph-based algorithms such as the Voronoi diagram algorithm [8], there are also A* algorithm [9], Probabilistic road maps algorithm [10, 11], rapidly-exploring random trees-based algorithm [12, 13]. Nevertheless, these algorithms rarely consider UAV kinematic and dynamic constraints, so they cannot be used in practical applications. In addition, these algorithms are dependent on cost maps, which must be developed and saved in advance, making the cost maps time-consuming to create. Another type of effective path-planning method is the potential fields-based method. Two classic instances of this type are the Artificial potential field algorithm [14] and interfered fluid dynamical system algorithm [15]. Such algorithms must globally establish the interaction between the attractive and repulsive fields to construct the flyable path for UAVs. Consequently, they are easily trapped in a local minima. Furthermore, sometimes it is impossible to guarantee a feasible path when the target and obstacles are too close.

It has been demonstrated that drone swarm path planning problem is an NP-hard problem, and the complexity of the problem grows with problem size [16]. To solve the NP-hard problems, meta-heuristics algorithms are effective and easy to implement.

The key challenge in dense swarm and environmental constraints is generating a collision-free path for drone swarm [17, 18]. In addition, deterministic approaches for building paths for drone swarm require a large amount of storage capacity and a long execution time [19].

Hence, to solve such a problem, proper optimization methods are necessary. Furthermore, optimization criteria may include the shortest path length, avoiding obstacles, shorter time missions, drone constraints (e.g., the amount of energy required to complete a mission, coverage area, etc.), and so on [20].

In recent years, population-based evolutionary algorithms have benefited greatly from advancements in swarm intelligence technology [21, 22], and they have a great capability to discover the optimal solution in an efficient and flexible manner. As a result, researchers are increasingly focusing on UAV path planning using these methods. A few of the most commonly used algorithms include Genetic Algorithm (GA) [23], Artificial Bee Colony (ABC) algorithm [24], Ant Colony Optimization (ACO) [25, 26], Differential Evolution (DE) [27, 28], Particle Swarm Optimization (PSO) [29], Spider Monkey Optimization (SMO) [30] etc.

The sine-cosine algorithm is one of the newly introduced swarm intelligence-based algorithm, which draws significant attention from the researchers because of its simplicity and ease of implementation in real-life applications. Mirjalili initially proposed this algorithm to solve optimization problems [31].

Over the last few years, several improved versions of SCA have been proposed. To enhance the exploitation ability of solutions and reduce the overflow of diversity present in the search equations of SCA, “an improved sine cosine algorithm for global optimization” was proposed in [32]. To enhance the exploration of the search space, the authors in [33], applied the opposition-based learning mechanism in SCA. The comparison results demonstrated that the proposed algorithm performs better than the original SCA and other considered meta-heuristic algorithms in terms of solving optimization problems. To effectively recognize the pathological brain in real-time, the authors in [34] combined an extreme learning machine with a modified sine cosine algorithm. Here the authors used the concept of mutation strategy in SCA to enhance the global search capability. Besides, a variety of meta-heuristic algorithms have been used to study the UAV path planning problems [35–37]. However, SCA has not been proposed for path planning for drone swarm in a 3D environment. This is due to its shortcomings of slow convergence and falling into local optimality when solving complex problems. Since the drone swarm path planning problem in a 3D environment is a very complex optimization problem, an appropriate path planning algorithm is required to be developed. To effectively plan the paths for the swarm of drones and overcome the disadvantages of the existing algorithms, such as frequently falling into local optimum solution and slow convergence, this paper proposes an "improved sine cosine algorithm" namely, iSCA. The main contributions of this paper are as follows.

- In iSCA, the chaos-based initialization of the population for better uniformity is used.
- It uses non-linearly decreasing step size to balance between local and global search process of SCA.
- The convergence factor is employed for faster convergence of SCA.
- The proposed iSCA is tested over drone swarm path planning problem and compared with other state-of-the-art algorithms.
- Applied the iSCA for tackling the 3D path planning problem for the drone swarm.

The remainder of this paper is arranged as follows: "Mathematical model for drone swarm path planning problem" describes the mathematical model for drone swarm path planning problem. The path planning algorithm based on the proposed iSCA is presented in "Path planning algorithm". "Simulation results and discussions" discusses the simulation results with a detailed comparison among the algorithms. Finally, the conclusion of this work is summarized in "Conclusion and future direction".

Mathematical Model for Drone Swarm Path Planning Problem

When planning the paths for drone swarm, it is important to consider some factors such as terrain area, the cost associated with each path, and drone's safety. The mission environment can have dangers like buildings, radars, mountains, or other impediments. In addition, the drone swarm consists of a large number of drones. Hence, objective functions must incorporate all these environmental factors as well as reflect their effects on performance. Drone swarm path planning problem is formulated as an optimization problem and then solved using the iSCA. Environmental restrictions and objective functions are covered in the ensuing sections.

Representations of Flying Area for Drone Swarm

In drone swarm path planning, the goal is to find an optimal and feasible path for the drones from their starting position to their target position under complex environmental constraints. Throughout this study, we refer to (x, y, z) as the three-dimensional coordinates of waypoints of the path. The flying spaces for drone swarm is expressed as follows [38].

$$\mathcal{S} = \left\{(x, y, z) | x_{lb} \le x \le x_{ub}, y_{lb} \le y \le y_{ub}, z_{lb} \le z \le z_{ub}\right\} \tag{1}$$

Where x_{lb}, y_{lb}, and z_{lb} are the lower limits of the flying space while x_{ub}, y_{ub}, and z_{ub} are the upper bounds.

Obstacle Model

Nowadays, it is possible to obtain accurate, up-to-date terrain maps and obstacles position using various sensing technologies such as infrared, LiDAR, GPS, etc. In this paper, it is assumed that the spatial boundaries and the locations of the obstacles are well known in advance. We model the obstacles as given in [36]. If $(x_{k_1}, y_{k_1}, z_{k_1})$ are the coordinates of the k_1th circular obstacle in a 3D environment with radius R_{k_1} then the k_1th obstacle can be represented as follows [39].

$$O_{k_1} = \left(x_{k_1}, y_{k_1}, z_{k_1}, R_{k_1}\right) \tag{2}$$

Where the coordinates $(x_{k_1}, y_{k_1}, z_{k_1})$ are calculated as follows:

$$x_{k_1} = R_{k_1} \cos(\theta) \sin(\phi) + x_c \tag{3}$$

$$y_{k_1} = R_{k_1} \sin(\theta) \sin(\phi) + y_c \tag{4}$$

$$z_{k_1} = R_{k_1} \cos(\phi) + z_c \tag{5}$$

Where (x_c, y_c, z_c) are the coordinates of center of the k_1th obstacle and $\theta \in [0, 2\pi], \phi \in \left[0, \pi/2\right]$.

Objective Function Modeling

In path planning, the objective function includes determining the length of the path, considering environmental constraints, and avoiding collisions with obstacles and other drones in the swarm. Our objective function aims to minimize the overall path length while avoiding obstacles. The objective function can therefore be expressed as follows [38].

$$F = F_{\text{pl}} + F_{\text{oc}} + F_{\text{mc}} \tag{6}$$

Where F_{pl} is the cost associated with path length, F_{oc} is the cost of drones collision with obstacles and F_{mc} is the collision cost among drones. The goal is to minimize the objective function F. The next subsection describes the mathematical formulations of F_{pl}, F_{oc}, and F_{mc}.

Cost Associated with Path Length

The expected flight path of a mission is a shorter one because shorter paths consume less fuel and are less likely to incur unforeseen threats. To evaluate the cost associated with path length, we use the following path length ratio (PLR) [40].

$$F_{pl} = \frac{\sum_{j=1}^{D-1} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2 + (z_{j+1} - z_j)^2}}{\sqrt{(x_D - x_1)^2 + (y_D - y_1)^2 + (z_D - z_1)^2}}. \quad (7)$$

Where D denotes the total number of waypoints in the path, (x_j, y_j, z_j) are the coordinates of the jth waypoint, (x_1, y_1, z_1) and (x_D, y_D, z_D) are the coordinates of the start and end waypoints of the path, respectively.

Here, the denominator represents the length of the shortest path between the start and the end waypoint while the numerator represents the length of the flight path. So, F_{pl} is always ≥ 1 and a smaller value of F_{pl} corresponds to a flight path with shorter length.

Obstacle Cost

To fly a drone safely, the planned path must avoid all obstacles. Even one point in the solution that passes through an obstacle may incur a high cost. If $path_i$ is the planned path for $drone_i$ and (x_j, y_j, z_j), $j = 1, 2, \dots, D$ are the coordinates of the jth waypoint in the $path_i$ then every waypoint $(j = 1, 2, \dots, D)$ of the $path_i$ should be checked against all obstacles to see if they fall into them. To do so, the distance between the waypoint's and the center of obstacles is taken into account. It is assumed that the waypoint does not fall into the obstacle if the distance between the waypoint and the center of the obstacle is greater than the radius of the obstacle. In this case, a negligible cost is given to the objective function as the obstacle cost. In contrast, when the distance between them is shorter than the radius of the obstacle, then the high cost is assigned as the penalty.

Thus, the cost for the obstacle avoidance is defined as follows [38].

$$F_{oc} = \sum_{j=1}^{D} \sum_{k_1=1}^{k} exp\left(-\frac{\alpha \times dist(j, k_1)}{R_{k_1}}\right) \quad (8)$$

where $\alpha \in [0, 1]$ is a control parameter, k is the total number of obstacles, R_{k_1} is the radius of k_1th obstacle and $dist(j, k_1)$ represents the distance between jth waypoint of the $path_i$ and center of the k_1th obstacle and is defined as follows:

$$dist(j, k_1) = \sqrt{(x(j) - x_0(k_1))^2 + (y(j) - y_0(k_1))^2 + (z(j) - z_0(k_1))^2} \quad (9)$$

where $(x(j), y(j), z(j))$ represents the coordinates of the jth waypoint and $(x_0(k_1), y_0(k_1), z_0(k_1))$ are the coordinates of the center of the k_1th obstacle.

Cost of Drone Member Collision

When planning the paths for drone swarm, collision avoidance must be considered. All drones should maintain a reasonable distance from one another. The probability of collision among drones increases as the drone swarm density increases space. It is therefore extremely important to ensure that drones are not too close to each other when drone swarm paths are generated. If $path_i$ is the planned path for $drone_i$ and $path_o$ is the planned path for any other drone then every waypoint's of $path_i$ must be checked with every waypoint's of other paths ($path_o$). To do so, it is necessary to consider a safety distance (sd) between paths.

The cost associated with collision among drones can be written as follows [38].

$$F_{\text{mc}} = \sum_{j=1}^{D} \sum_{j_1=1}^{\tilde{D}} \exp\left(-\frac{\alpha \times dist(j, j_1)}{sd}\right) \quad (10)$$

where D and $\tilde{D}$ are the number of waypoints in the $path_i$ and $path_o$, respectively. $\alpha \in [0, 1]$ is a control parameter, sd is the inter-drone distance, and $dist(j, j_1)$ represents the distance between $path_i$ and $path_o$ and is defined as follows:

$$dist(j, j_1) = \sqrt{(x_j - x_{j_1})^2 + (y_j - y_{j_1})^2 + (z_j - z_{j_1})^2} \quad (11)$$

where (x_j, y_j, z_j) and $(x_{j_1}, y_{j_1}, z_{j_1})$ are the waypoints of $path_i$ and $path_o$, respectively.

In the above model, the cost of member collisions among drones is mostly driven by the distance between $path_i$ and other paths ($path_o$). As the safety distance (sd) should be maintained, the cost will increase when the distance between paths is $\leq sd$, and it decreases rapidly as the distance between the paths increases.

Path Planning Algorithm

Sine Cosine Algorithm (SCA)

The SCA is a new population-based meta-heuristic algorithm developed by Mirjalili [31], which utilizes a set of candidate solutions for performing the search. This is a method whereby guided randomness is created through the use of sine and cosine trigonometric functions. In SCA, the global solution is called a destination point and the solution vectors are called candidate solutions.

For the drone swarm path planning problem, let each feasible path represent the feasible candidate solution of the population in SCA. Assuming that the number of waypoints for each candidate solution is D, then for the three-dimensional path planning problem, the waypoints of the $path_i$ (ith candidate solution) can be expressed as follows:

$$X_i = (x_{i,1}, \dots, x_{i,D})^{\mathrm{T}} \tag{12}$$

where $x_{i,j}$, $j \in 1, \dots, D$ represents the jth waypoint of the ith candidate solution and are denoted as follows:

$$x_{i,1} = (x^x_{i,1}, x^y_{i,1}, x^z_{i,1}) \tag{13}$$

$$\dots \tag{14}$$

$$x_{i,D} = (x^x_{i,D}, x^y_{i,D}, x^z_{i,D}) \tag{15}$$

$(x^x_{i,j}, x^y_{i,j}, x^z_{i,j})$, $j \in 1, \dots, D$ represents the coordinates of jth waypoint of the ith candidate solution in a three-dimensional space.

Thus if N denote the total number of candidate solutions then the population (swarm) can be represented as

$$P = (X_1, X_2, \dots, X_N)^T \tag{16}$$

For SCA with N number of candidate solutions, there is one destination point (global best solution), which can be written as

$$G_{best} = (g_{best,1}, \dots, g_{best,D}) \tag{17}$$

Now, in tth iteration the position of each candidate solution is updated based on the following formula [31]:

$$X^{(t+1)}_{i,j} = \begin{cases} X^{(t)}_{i,j} + r_1 \times \sin(r_2) \times \mid r_3 \times G^t_{best} - X^t_{i,j} \mid, & \text{if } r_4 < 0.5 \\ X^{(t)}_{i,j} + r_1 \times \cos(r_2) \times \mid r_3 \times G^t_{best} - X^t_{i,j} \mid, & \text{Otherwise} \end{cases} \tag{18}$$

Where r_2, r_3, and r_4 are random numbers in the ranges $(0, 2\pi)$, (0, 1), and (0, 1), respectively. Here, the parameter r_4 is known as the switching parameter because it is used to choose the search paths using the sine or cosine function.

The parameter r_1 is known as the control parameter, which decreases linearly from a number β to 0. It is responsible to manage the exploration and exploitation during the search by changing its value. $r_1 > 1$ indicates the exploration of the search space, while $r_1 < 1$ indicates exploitation. r_1 is defined as follows:

$$r_1 = \beta \times \left(1 - \frac{t}{\mathrm{Max}_{\mathrm{iteration}}}\right) \tag{19}$$

Where t and $\mathrm{Max}_{\mathrm{iteration}}$ are the current iteration number and the maximum number of iterations, respectively.

The pseudo-code of SCA is shown in Algorithm 1.

Improved Sine Cosine Algorithm (iSCA)

Despite the fact that the original SCA has enough exploration capability to diversify the search space, it often gets stuck in local optima and undergoes premature convergence when tackling complex problems [41]. Drone swarm path planning problem is a complex problem. It needs an efficient algorithm that balances the exploration and exploitation, efficiently. Therefore, it is essential to balance exploration and exploitation in SCA when performing the search operations to find the optimal path.

To prevent trapping in local optima and to search more accurately and rapidly for global optima, the present study proposes improvements in the SCA as follows:

Algorithm 1 Pseudo-code of SCA

```
Initialize the $N$ number of candidate solutions randomly.
Evaluate all the solutions.
Identify the best candidate solution.
while $t < Max_{iteration}$ do
    Generate $r_1$ using (19).
    for $i = 1 : N$ do
        for $j = 1 : D$ do
            Get the values of $r_2$, $r_3$ and $r_4$.
            Update the position of each candidate solution using (18).
        end for
    end for
    Evaluate the updated population.
    Identify the best candidate solution.
end while
```

- Chaos-based initialization of candidate solutions.
- Better position update strategy by introducing non-linearly decreasing step size.
- Incorporation of the convergence factor in the search mechanism to speed up the convergence rate.

Chaos-Based Population Initialization

The population initialization in any evolutionary algorithm plays a very important role in the convergence speed and quality of the final solution. In general, random initialization is the most commonly used method of generating initial population in the absence of any information about the solution. The SCA uses uniformly distributed random solutions to initialize the population of candidate solutions. According to [42], when the distribution is more uniform, the population maintains rich diversity, which increases the chance of faster convergence and better solution quality. Hence, chaos-based initialization contributes in maintaining better diversity among the potential drone swarm paths. Logistic maps have the advantage of a more uniform distribution when compared with random distribution over 10,000 times [42].

In this work, to enrich the diversity of the initial population, the logistic map, which is one of the simplest and the most widely used chaotic map, is used [43].

$$y_{j+1} = \mu \times y_j \times (1 - y_j), \quad j = 0, 1, 2, \dots \tag{20}$$

Where y_j is the jth chaotic variable. μ is the bifurcation coefficient. A chaotic state occurs if $\mu \in [3.57, 4]$. When $\mu = 4$, $y_0 \in (0, 1)$, the system produces a uniform chaotic signal, which will be employed for the initialization of the candidate solutions.

Steps to implement logistic map-based initialization are as follows:

1. First, set $y_0 \in (0, 1)$ and generate D (population dimension) chaotic variables using following Eq. (21).
$$y_{j+1} = \mu \times y_j \times (1 - y_j), \quad j = 0, 1, 2, \dots, D \tag{21}$$
Where y_j denotes the jth variable.
2. Repeat step 1 for $i = 1, 2, \dots, N$ (population size), and generate the initial chaotic variables for each candidate solution i.
3. Initialize the candidate solutions as follows:
$$x_{i,j} = x_{\min,j} + y_{i,j} \times (x_{\max,j} - x_{\min,j}), \quad i = 1, 2, \dots, N,\ j = 1, 2, \dots, D \tag{22}$$
Where $x_{\max,j}$ and $x_{\min,j}$ are the upper and lower bounds of the jth variable, respectively.
4. Finally, ith candidate solution using logistic map is
$$X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D}); \ \forall i = 1, 2, \dots, N \tag{23}$$

Pseudo-code of chaos based population initialization is presented in Algorithm 2.

Algorithm 2 Chaos based population initialization

```
Input: Population size ($N$), population dimension ($D$), bifurcation coefficient ($\mu$) & $y_0 = rand$.
for $i = 1 : N$ do
    for $j = 1 : D$ do
        Generate the chaotic variables using (21).
    end for
end for
for $i = 1 : N$ do
    for $j = 1 : D$ do
        Generate the initial candidate solution using (22).
    end for
end for
Output: $N$ initial solutions with $D$ number of variables.
```

Improved Position Updating Mechanism

In SCA, the control parameter r_1 controls exploration in the early iteration and exploitation in the later iteration. This transition parameter can be further modified to balance the exploration and exploitation of the search process. The control parameter r_1 in SCA is a linear function that decreases linearly from β to 0. Because of its linearity, sometimes it creates abrupt changes in jumping from one iteration to the next. In some cases, abrupt changes may result in the skipping of good solutions; thus, valuable information about the quality of search areas might be lost. In this study, a modified formula for r_1 so that it decreases exponentially from γ to 0 is proposed to avoid all these issues.

$$r_{1_{\text{new}}} = \gamma \times exp\left(-\frac{t^2}{(\beta \times \text{MAX}_{\text{iteration}})^2}\right) \tag{24}$$

where t and $\text{MAX}_{\text{iteration}}$ are current and maximum number of iterations, respectively. γ is a user-defined parameter.

Convergence Factor

Further, a convergence factor CF is employed in the search mechanism. This convergence factor CF helps iSCA to converge rapidly while balancing exploration and exploitation. CF is defined as follows:

$$\text{CF} = \beta \times \left(1 - \frac{t}{\text{MAX}_{\text{iteration}}}\right) \tag{25}$$

In Eq. (25), the convergence factor (CF) is inversely proportional to the number of iterations. Its small value corresponds to less dependency over the current position, and its higher value plays more role of the current position in deciding the new position. Initially, when CF is large, the search process is significantly guided by the current position, and in the later iteration, when CF is small, it plays less role, and the new position is more depends upon the global best solution.

As mentioned above, the non-linearly decreasing step size (Eq. 24) helps in balancing exploration and exploitation of the search process very well, while the convergence factor (Eq. 25) helps in fast convergence. Thus the following proposed search Eq. (26) is used in iSCA which merges both techniques, cancels the absolute value term, to obtain better performance in drone swarm path planning in terms of solution quality, accuracy, and convergence speed.

$$X_{i,j}^{(t+1)} = \begin{cases} \text{CF} \times X_{i,j}^{(t)} + r_{1_{\text{new}}} \times \sin(r_2) \times \left(r_3 \times G_{\text{best}}^t - X_{i,j}^t\right), & \text{if } r_4 < 0.5 \\ \text{CF} \times X_{i,j}^{(t)} + r_{1_{\text{new}}} \times \cos(r_2) \times \left(r_3 \times G_{\text{best}}^t - X_{i,j}^t\right), & \text{Otherwise} \end{cases} \tag{26}$$

Symbols have their usual meaning. The pseudo-code of improved SCA (iSCA) for drone swarm path planning is shown in Algorithm 3.

Algorithm 3 The improved SCA (iSCA) for drone swarm path planning

```
Environment construction:
Set the boundary for drones flying area.
Set obtacles positions.
Set the starting and destination position for each drone.
for $d = 1$ : total number of drones do
    Initialize the $N$ number of candidate solutions using chaos based
    initialization (Equation (20)).
    Evaluate all the solutions.
    Identify the best candidate solution.
    Main loop:
    while $t < Max_{iteration}$ do
        Calculate $r_{1_{new}}$ using (24) .
        Calculate $CF$ using (25).
        for $i = 1 : N$ do
            for $j = 1 : D$ do
                Generate $r_2$, $r_3$ and $r_4$.
                Update the position of each candidate solution using (26).
            end for
        end for
        Evaluate the updated population.
        Identify the best candidate solution.
    end while
    Output: The optimal path for drone $d$.
end for
```

Table 1 Drones starting and destination positions

Drones	Starting position	Destination position
Drone1	(200, 800, 350)	(16000, 4000, 350)
Drone2	(200, 3800, 350)	(16000, 7000, 350)
Drone3	(200, 6800, 350)	(16000, 10000, 350)
Drone4	(200, 9800, 350)	(16000, 13000, 350)
Drone5	(200, 12800, 350)	(16000, 16000, 350)

Table 2 Obstacles position

Obstacles	Positions	Radius
Obstacle1	(5000, 10000, 0)	1800
Obstacle2	(10000, 2000, 0)	1200
Obstacle3	(10000, 8000, 0)	1100
Obstacle4	(5000, 2000, 0)	1500

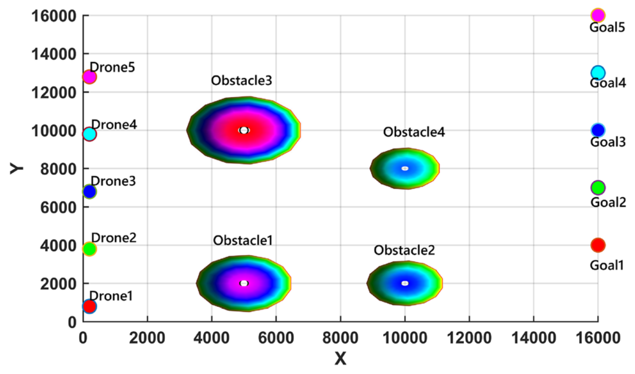


Fig. 1 Two-dimensional view of the obstacles, starting and destination points of the drone swarm

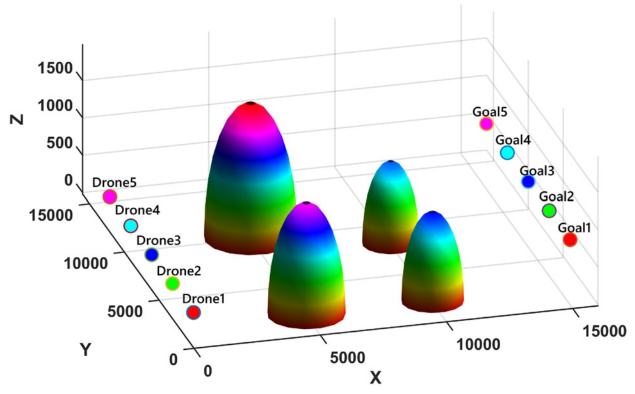


Fig. 2 Three-dimensional view of the obstacles, starting and destination points of the drone swarm

Table 3 Parameters setting for all algorithms

Algorithms	Parameters
SCA	Same as in [31]
RCN	$\beta = 2, \gamma = 1$
CL	$\beta = 2, \mu = 4$
iSCA	$\beta = 2, \gamma = 1$
PSO	$w_{\max} = 0.8, w_{\min} = 0.4, c_1 = 1.47, c_2 = 1.47$
IPSO	$w_{\max} = 0.9, w_{\min} = 0.4, c_1 = 1.47, c_2 = 1.47, \epsilon_{\max} = 0.9, \epsilon_{\min} = 0.05, V_{max} = 0.3$
ABC	Same as in [24]

Table 4 Common parameters

Population size (N) = 300
Maximum iteration = 150
Safety distance (sd) = 80m
Number of waypoints (D) = 20
$\mu = 4$

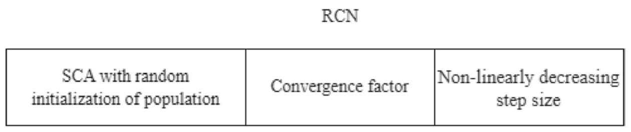

RCN
SCA with random initialization of population
Convergence factor
Non-linearly decreasing step size

Fig. 3 Framework of RCN

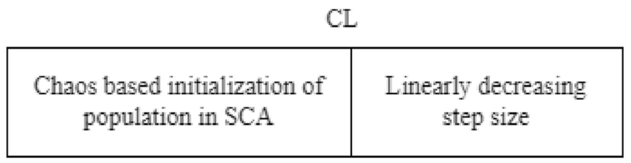

CL
Chaos based initialization of population in SCA
Linearly decreasing step size

Fig. 4 Framework of CL

Time Complexity of the Proposed Algorithm

In the process of initializing the population, N number of candidate solutions are generated, and each candidate solution is a D dimensional vector in a 3D environment. If $\text{Max}_{\text{iteration}}$ denotes the maximum number of iterations and d_{drones} denotes the size of the swarm then the time complexity to solve the path planning problem in a 3D environment using any population-based optimization algorithm is $\mathcal{O}(N * D * \text{Max}_{\text{iteration}} * d_{\text{drones}})$. Therefore it is clear that the

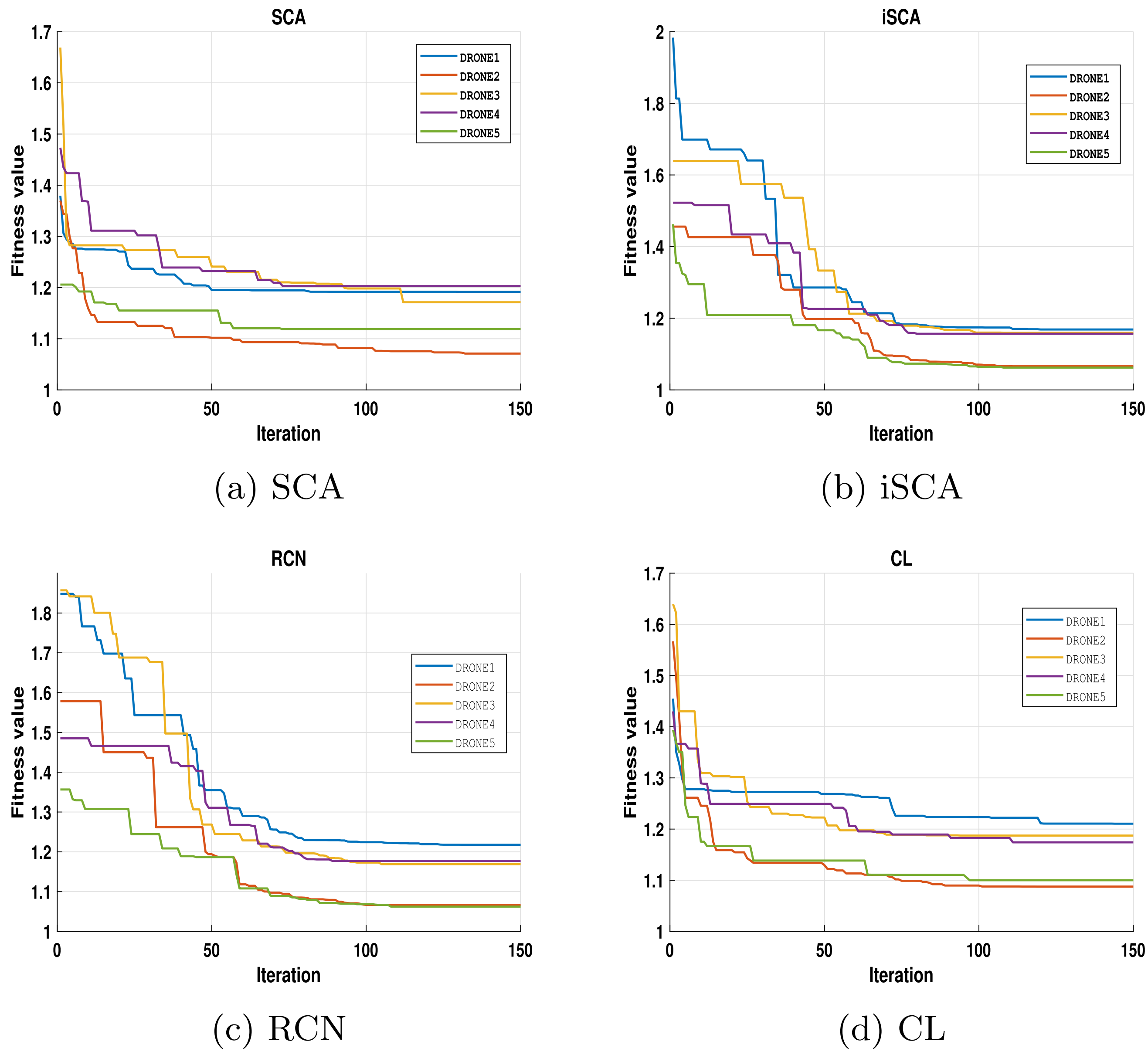


Fig. 5 Fitness value over iteration for drone swarm path formation

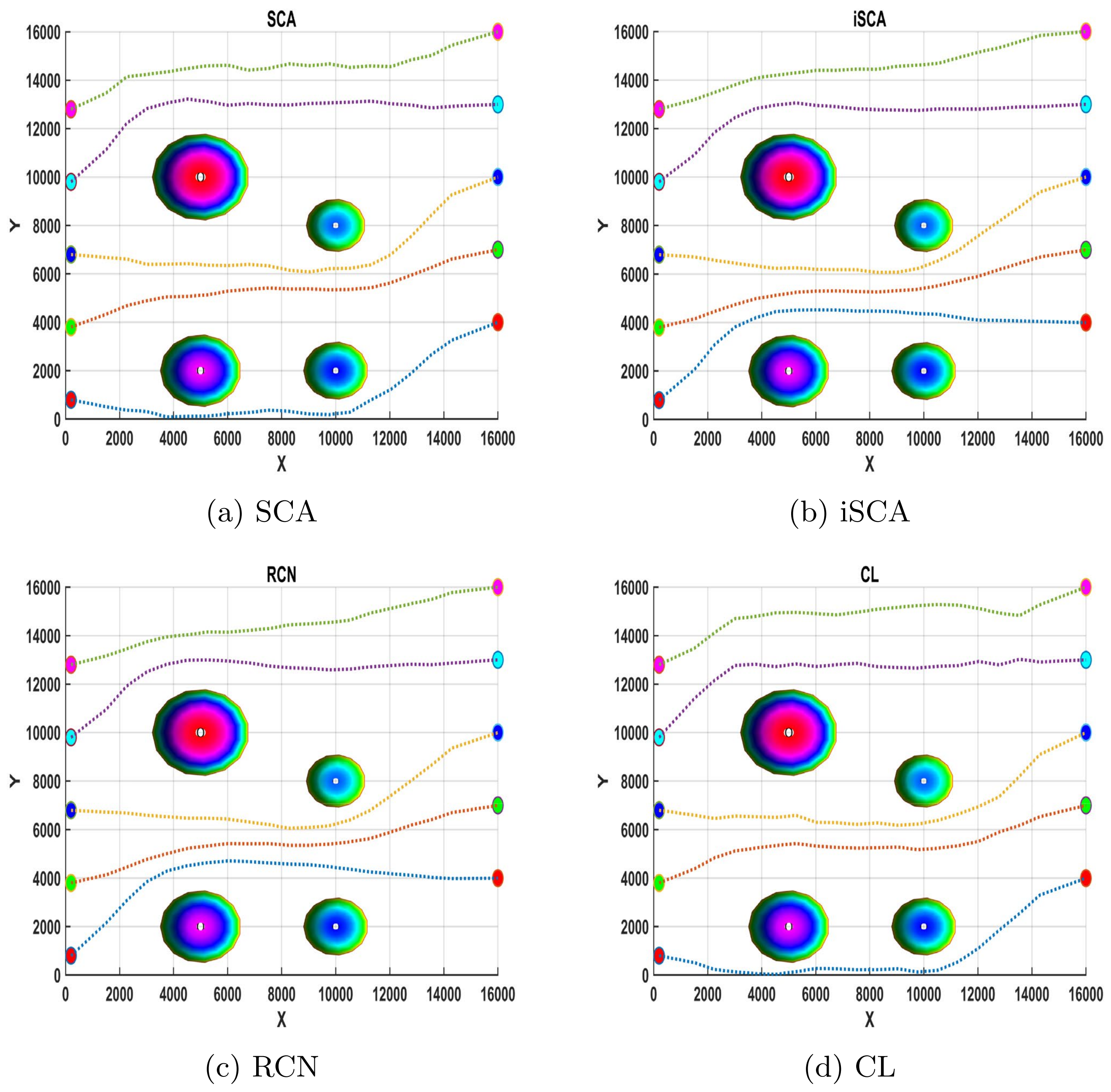
modifications in the proposed iSCA over SCA are not adding any time complexity in solving the considered problem.

Simulation Results and Discussions

In this section, simulation results and comparisons are presented in order to show the performance of the proposed iSCA over drone swarm path planning problem. The experiments are carried out in a MATLAB environment on a server with a 3.70 GHz CPU, 64 GB of RAM, and a 64-bit operating system.

Parameter Settings

Parameter settings play an important role in the performance of an algorithm as appropriate parameters may lead to better results of the algorithm. In the simulation environment, five drones are assumed to fly, simultaneously from their starting position to their destination position. Table 1 shows the positions of the starting and destination positions of each drone in a 3D space of size $16000 \times 16000 \times 16000$. Four static obstacles are placed in the search space, whose positions are listed in Table 2. The location of each obstacle and the current positions of drones with their destinations (goals) are



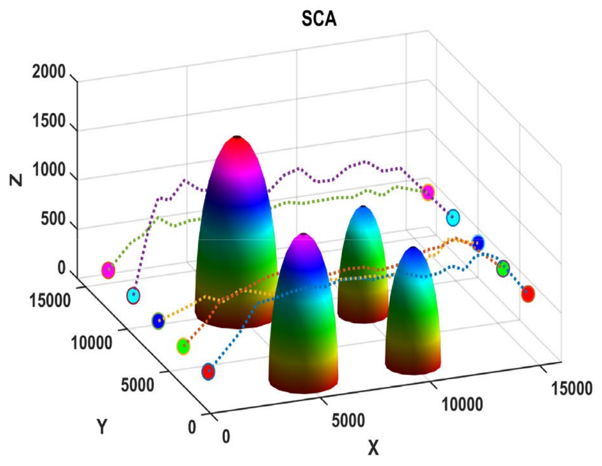
(a) SCA (b) iSCA (c) RCN (d) CL

Fig. 6 2D view of the planned path for drone swarm

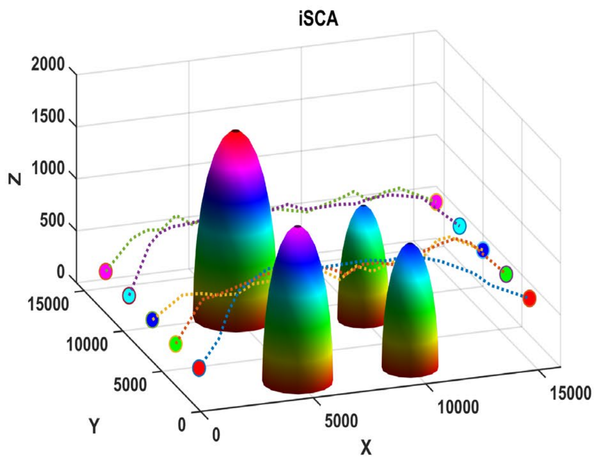
presented in 2D and 3D views in Figs. 1 and 2, respectively. The parameters corresponding to all the considered algorithms are presented in Table 3. The parameters in Table 4 are common to all the algorithms. In Table 4, *sd* stands for safety distance for collision avoidance among the drones, *D* stands for the total number of waypoints, *N* represents the population size, and μ is the bifurcation coefficient.

Results and Comparisons

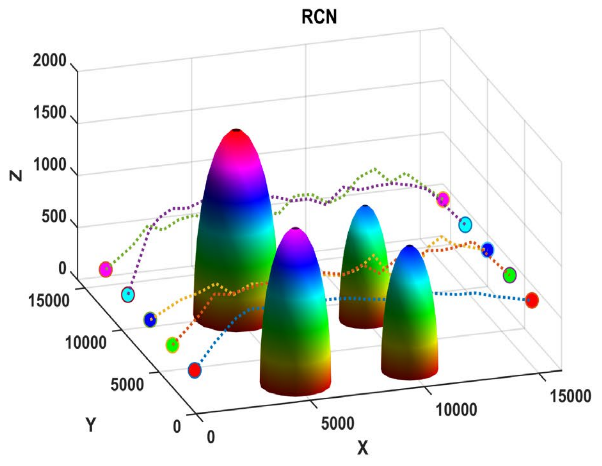
This section examines the effectiveness of the proposed iSCA by taking into account a number of performance metrics, including drone swarm formation running time, failure and success rates, convergence speed, and solution quality. The corresponding subsections contain an analysis and record of the outcomes from the algorithms under consideration.



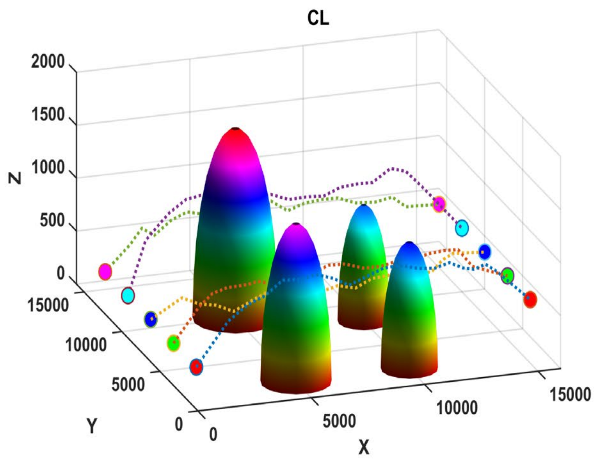
(a) SCA



(b) iSCA



(c) RCN



(d) CL

Fig. 7 3D view of the planned path for drone swarm

Table 5 Average fitness value of the formation

Algorithm	Drone1	Drone2	Drone3	Drone4	Drone5	FAFV	Improved %
SCA	1.150535239	1.045553413	1.111839756	1.141272442	1.063580563	1.102556283	NA
iSCA	**1.125486909**	1.038086312	**1.097011195**	**1.111504654**	**1.026271119**	**1.079672038**	2.076
RCN	1.133059711	**1.037975431**	1.103182588	1.119762864	1.028719366	1.084539992	1.634
CL	1.150075141	1.048581918	1.118792639	1.139484139	1.061501466	1.103687061	−0.103

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

Comparison with Original SCA

In this subsection, the performance of the proposed iSCA is compared with the original SCA. Since iSCA is proposed with two important inclusions, that is, convergence factor and non-linearly decreasing step size, so to examine the significance of each modifications, we additionally considered both factors, independently for the comparison. Following two variants of SCA are considered using each modification independently for the comparison.

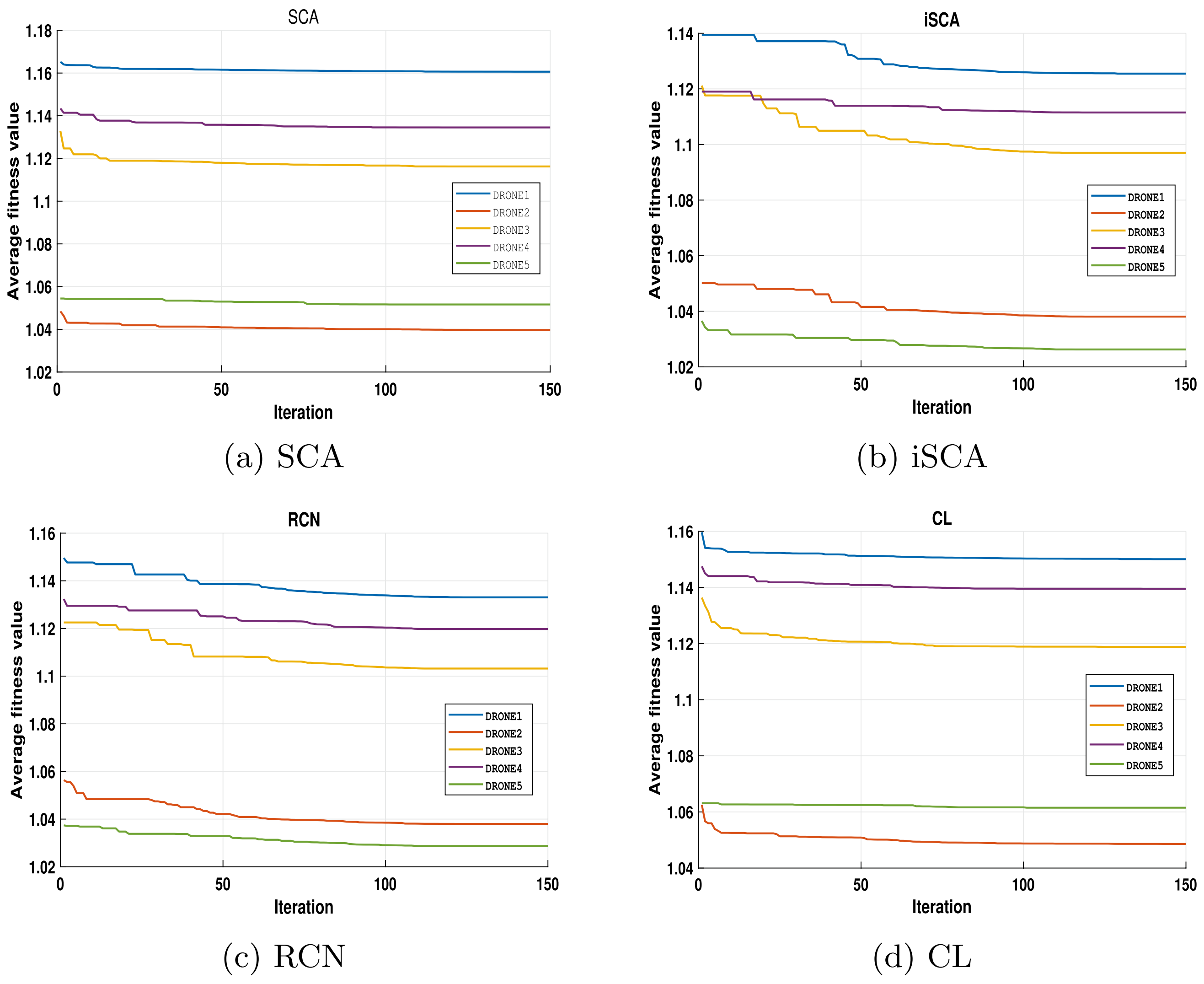


Fig. 8 Average fitness value over iteration on 40 runs for drone swarm path formation

Table 6 Average minimum iterations of the formation

Algorithm	DRONE1	DRONE2	DRONE3	DRONE4	DRONE5	FAMI
SCA	25	38	**37**	**31**	40	**34.2**
iSCA	38	78	111	37	50	62.8
RCN	**22**	**28**	61	41	41	38.6
CL	29	45	57	38	**21**	38

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

- RCN: This algorithm is formed by including uniformly distributed population initialization along with non-linearly decreasing step size and convergence factor in the original SCA as shown in Fig. 3.
- CL: This algorithm initializes the population using a chaos map and linearly decreasing step size in SCA as shown in Fig. 4.

The performances over drone swarm path planning problem of SCA, iSCA, RCN, and CL are discussed and comparative

Table 7 Failure rate of the formation

Algorithm	DRONE1	DRONE2	DRONE3	DRONE4	DRONE5	AFN	FR	% improvement
SCA	40	**0**	4	25	**0**	13.8	34%	NA
iSCA	**9**	**0**	**1**	**1**	**0**	**2.2**	5.5%	84.057
RCN	16	**0**	3	5	**0**	4.8	12%	65.217
CL	40	**0**	9	27	**0**	15.2	38%	−10.144

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

Table 8 Formation running time by different algorithms

Algorithm	Formation running time (seconds)	% improvement
SCA	1.482247	NA
iSCA	**1.163345**	21.5147
RCN	**1.163345**	21.5147
CL	1.467294	1.0088

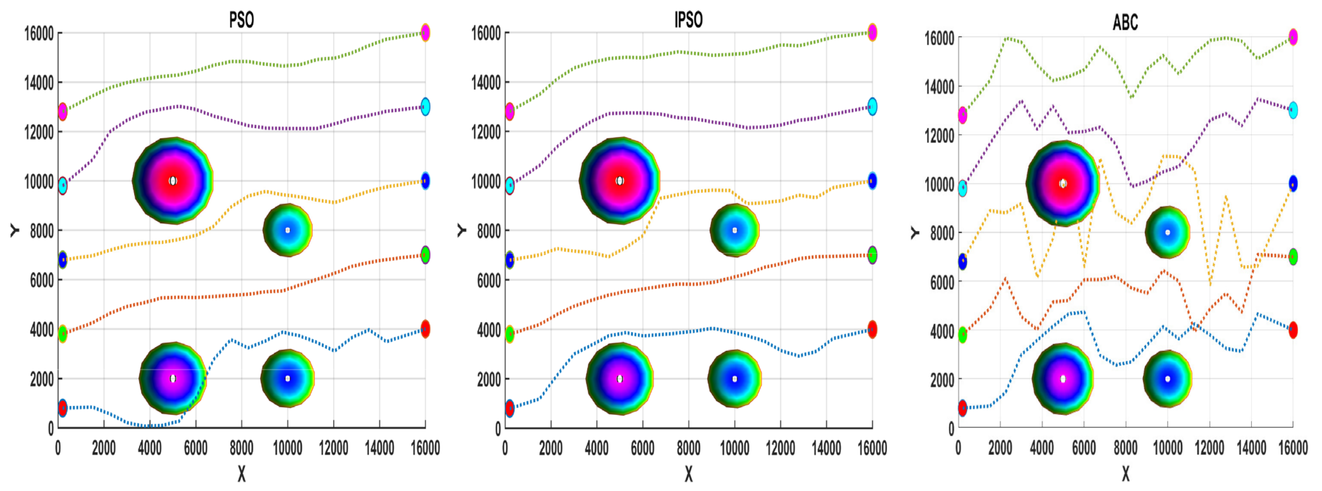
Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

study has been carried out based on the performance indicators given below:

Comparison based on solution quality: The solution quality can only be measured through objective function value (fitness value (FV)). Firstly, we have recorded the fitness values of all the considered algorithms over all the iterations in a single run. Figure 5a–d show the graphical representation of fitness values over iterations of all the algorithms considered in this subsection. From these results, it is clear that the FV of the proposed iSCA outperforms SCA, RCN, and CL. This comparison justifies that both the modifications along with chaos-based initialization is necessary to achieve this superior performance of iSCA. In other words, all three modifications in SCA are jointly responsible for better performance of iSCA.

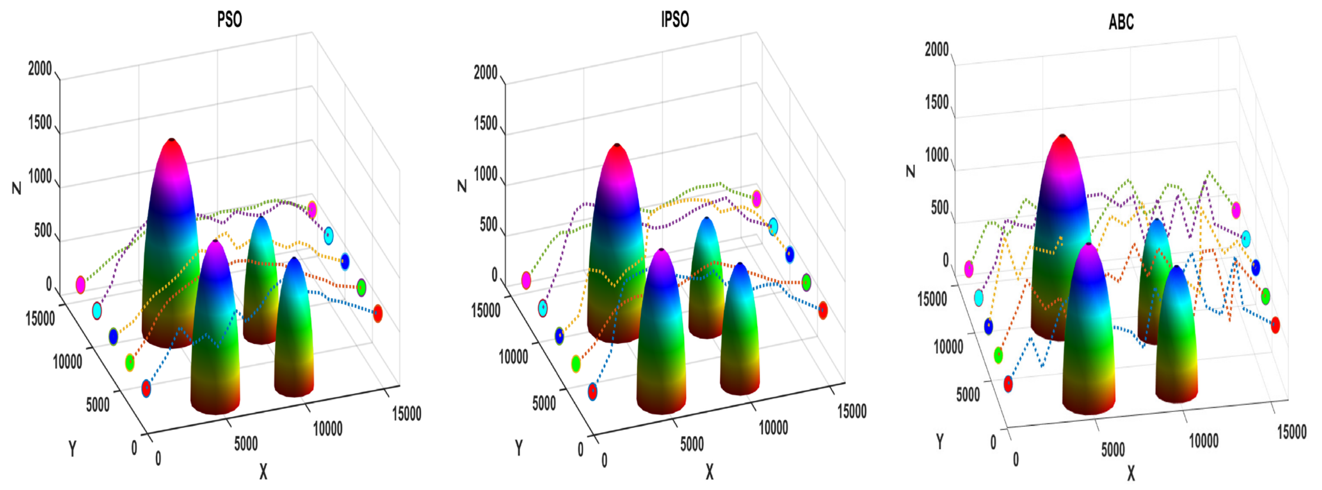
Since randomness is present in all the considered algorithms so it is not enough to take a decision through a single run. Thus in order to do a fair comparison among algorithms, we use the Monte-Carlo simulations with 40 runs to each algorithms and analysed the results. Figure 6a–d show the planned paths for drones by SCA, iSCA, RCN, and CL, respectively in 2D views. While Fig. 7a–d represent the 3D views of the planned path for drones for considered algorithms. From Figs. 6a–d and 7a–d, it can be seen that the planned path for each drone by each algorithm is obtained without collision with obstacles and among drones. Thus it is guaranteed that all the algorithms SCA, iSCA, RCN, and CL can generate a feasible path for each drone. Table 5 shows the average fitness value (AFV) of the formation during Monte-Carlo simulations. The best results are highlighted with boldface. Figure 8a–d shows the iteration-wise average fitness value (AFV). It can be observed from Table 5 that from the single drone perspective, iSCA has outperformed all the algorithms SCA, RCN, and CL for drones 1, 3, 4, and 5 but in the case of drone 2, iSCA no longer outperforms RCN. However, it is reasonable that the AFV of the formation is more important and fair than the individual drone's performance. Because the AFV of the formation indicates the overall solution quality of an algorithm, as well as the safety and cost-effectiveness of drone operations in the flying environment. Moreover, AFV for a single drone can easily be affected by different environmental constraints, so it is not good enough to evaluate the performance of an algorithm from a single drone's perspective in the case of drone swarm optimal path planning. The AFV of the formation in Table 5 shows that iSCA has the least AFV, RCN performs second, and CL worst. As compared to original SCA, the AFV of iSCA, RCN, and CL has decreased to 2.076%, 1.634%, and − 0.103%, respectively. Note that CL is not able to outperform SCA while RCN outperforms SCA independently, but RCN and CL which forms iSCA outperforms SCA as shown in Table 5. Overall, iSCA has a better solution quality than SCA, RCN, and CL.

Comparison based on convergence speed: The convergence speed is an important performance indicator in analyzing the effectiveness of the algorithm. Figures 5a–d and 8a–d are the convergence curves of SCA, iSCA, RCN, and CL in single and multiple runs, respectively. To show the evolution process of the algorithms over iterations, we assume that the *x*-axis represents the iteration number and *y*-axis represents the average fitness value in these Figures. In addition, we also adopt the minimum number of iterations that requires to reach the optimal solution as another indicator to evaluate the convergence speed. In this process, firstly, we set the criteria for the feasible solution as $|AFV_{t-20} - AFV_t| < 0.001$. In other words, $|AFV_{t-20} - AFV_t| < 0.001$ represents the difference in AFV obtained in 20 consecutive iterations. Here, t represents the current iteration number and AFV represents the average fitness value that is obtained in 40 runs. Table 6 represents the average minimum iterations (AMIs) of the formation for each of the considered algorithms. It can be



(a) PSO (b) IPSO (c) ABC

Fig. 9 2D view of the planned path for drone swarm



(a) PSO (b) IPSO (c) ABC

Fig. 10 3D view of the planned path for drone swarm

Table 9 AFV of the formation

Algorithm	Drone1	Drone2	Drone3	Drone4	Drone 5	FAFV
iSCA	**1.125486909**	1.038086312	**1.097011195**	**1.111504654**	**1.026271119**	**1.079672038**
PSO	1.274060312	1.031918232	1.150279379	1.137454152	1.039364774	1.12661537
% improved in iSCA	11.6613	− 0.5977	4.6309	2.2813	1.2598	4.1667
IPSO	1.167843329	**1.030065184**	1.152098958	1.116741579	1.039040203	1.10115785
%improved in iSCA	3.6274	− 0.7786	4.7815	0.4689	1.229	1.9512
ABC	1.298236247	1.163505086	1.15079026	1.466059323	1.103041915	1.236326566
%improved in iSCA	13.3064	10.7794	3.5704	35.4554	6.9599	12.671

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

Table 10 Failure rate of the formation

Algorithm	DRONE1	DRONE2	DRONE3	DRONE4	DRONE5	AFN	FR	% improved in iSCA
iSCA	**9**	**0**	**1**	**1**	**0**	**2.2**	5.5%	NA
PSO	40	**0**	35	21	**0**	19.2	48%	60.416
IPSO	40	**0**	40	**0**	**0**	16	40%	86.25
ABC	40	27	25	40	8	30	55%	92.142

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

Table 11 Average minimum iterations of the formation

Algorithm	DRONE1	DRONE2	DRONE3	DRONE4	DRONE5	FAMI
iSCA	38	78	111	37	50	62.8
PSO	48	77	30	44	68	53.4
IPSO	45	42	52	57	32	45.6
ABC	**21**	**21**	**21**	**21**	**21**	**21**

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

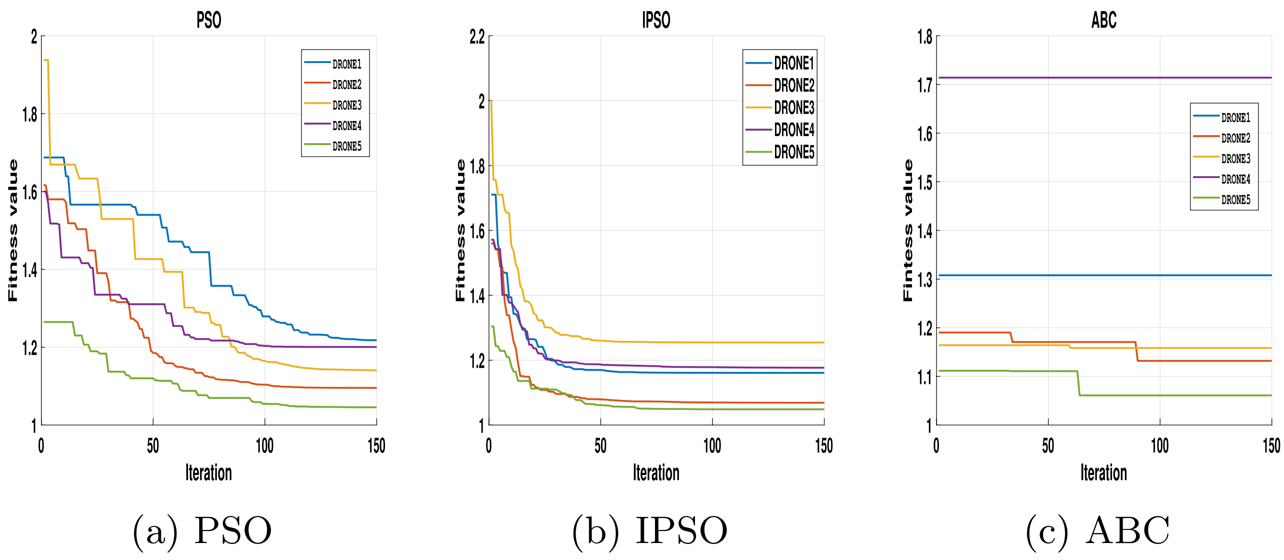


Fig. 11 Fitness value over iteration for drone swarm path formation

seen from Table 6 that SCA has the least AMI with a value of 34.2, while iSCA has the largest AMI with a value of 62.8. On the other hand, CL and RCN are 2nd and 3rd in the list with values 38 and 38.6, respectively. Though SCA obtained the highest convergence speed than iSCA, RCN, and CL, but it can be observed that the solution obtained by SCA with this fast convergence speed is sub-optimal and inferior than iSCA, RCN, and CL. This means that SCA can easily get trapped to local optima. Since the non-linearly decreasing step-size along with convergence factor are present in iSCA, these factors help iSCA to avoid stagnation and provide the searching ability for more iterations to obtain global optima.

Comparison based on success and failure rates: Besides solution quality and convergence speed, success and failure rate is also another important factor in analysing the performance of the considered algorithms. This indicator measures the reliability of the algorithm. In this work, Monte-Carlo simulations are carried out with 40 runs, and failure number (FN) is defined as the number of times the

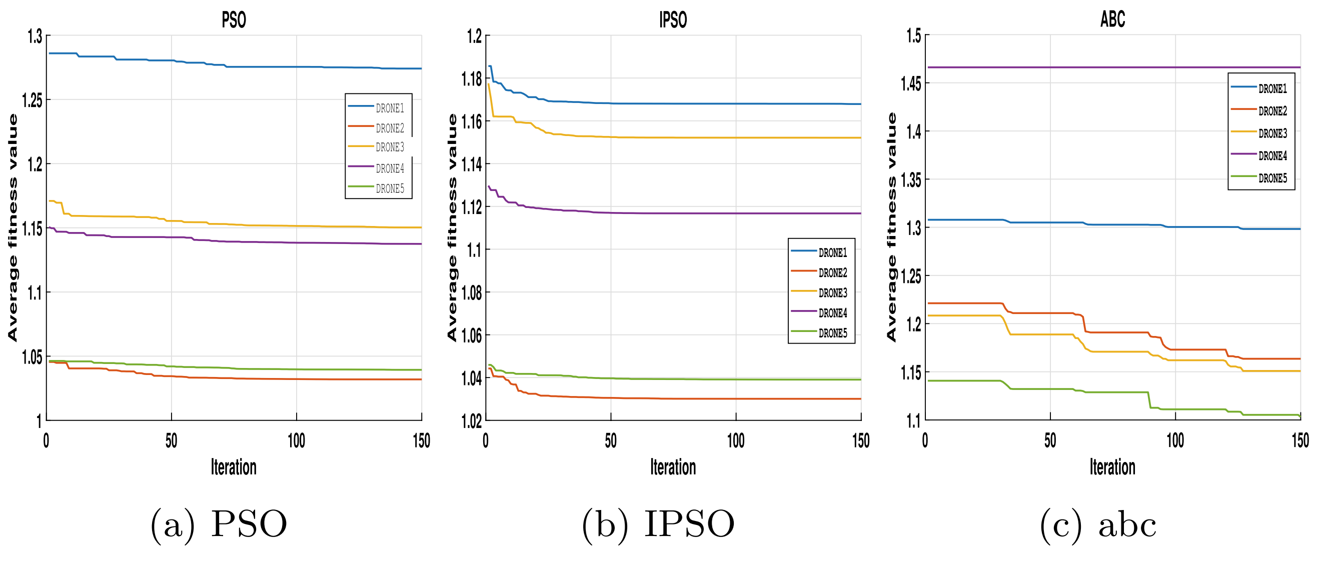


Fig. 12 Average fitness value over iteration on 40 runs for drone swarm path formation

Table 12 Formation running time of different algorithms

Algorithm	Formation running time (seconds)	% improved in iSCA
iSCA	**1.163345**	NA
PSO	1.18559	1.8762
IPSO	1.45934	20.2827
ABC	1.45934	20.2827

Bold values represent the best results that the associated algorithm produced when compared to all other algorithms that were taken into consideration

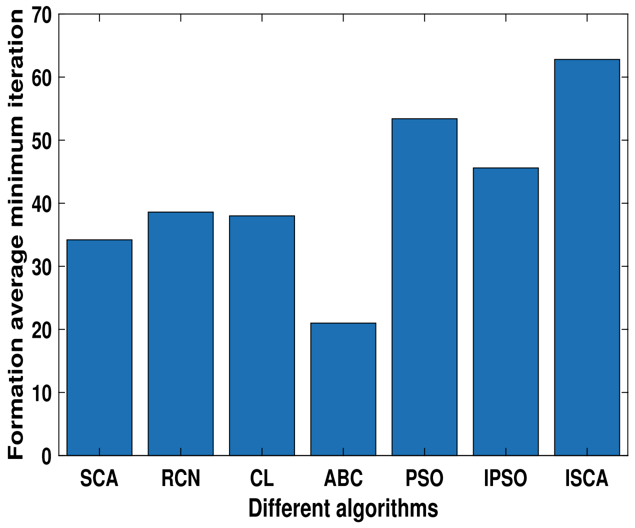


Fig. 13 Formation average minimum iterations by different algorithms

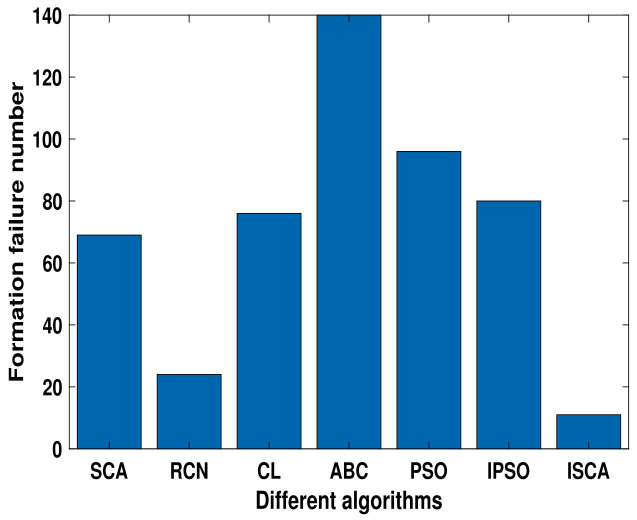
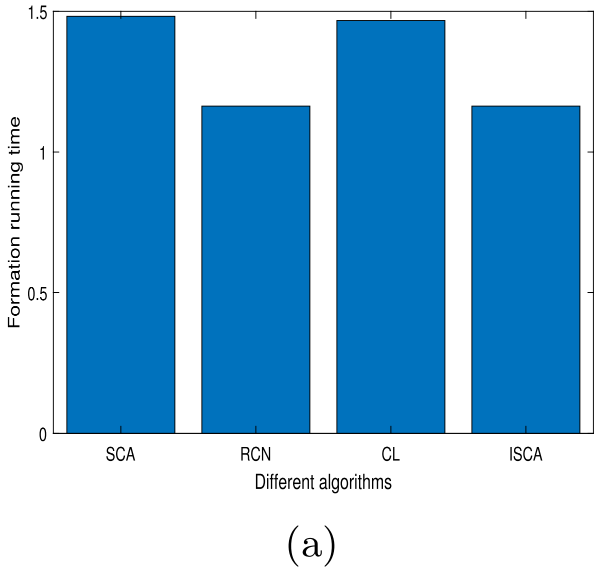


Fig. 14 Formation failure number by different algorithms

final fitness value ($FV_{t=150}$) is greater than or equal to 1.13. This means that the path generated by the algorithm is too long, and/or the drone collides with an obstacle, and/or the collision happens among drones. On the other hand, a zero failure rate indicates that $FV_{t=150} < 1.13$. The average failure number (AFN) of the formation and the accumulated path failure rate (FR) in the Monte-Carlo simulations on 40 runs are presented in Table 7. Figure 14 is the graphical representation of the formation failure number (FN) of considered algorithms. From Table 7, one can see that the proposed iSCA has the least failure rate with an improved percentage of 84.057 as compared to the original SCA, while 65.217% improvement is obtained by RCN. On the other hand, there



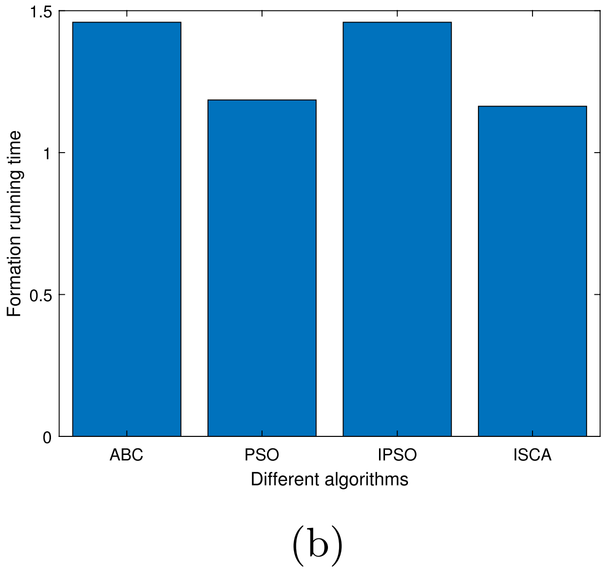


Fig. 15 Formation running time by different algorithms

is no improvement in CL. This shows that the proposed iSCA can achieve the highest success rate or reliability in path generation and guarantees a significant improvement in solution quality as compared to SCA, RCN, and CL.

Comparison based on running time: A drone swarm path planning and collision avoidance depend heavily on running time, which is another crucial component to consider when assessing an algorithm's performance. The running time is the time required to complete the task from the starting position to the destination position of the drone swarm. The results are presented in Table 8 and Fig. 15a. From Table 8, one can see that the increased percentage of the iSCA, RCN, and CL as compared to the original SCA are 21.5147, 21.5147, and 1.0088, respectively. These results demonstrate the ability of the proposed iSCA to generate the optimal paths for drone swarm in a faster way.

Comparison with Other Algorithms

In this subsection, we have compared the proposed iSCA with other meta-heuristics such as PSO [29], ABC [24], and one of the recently proposed improved PSO (IPSO) [38], which increases the solution quality and convergence speed of basic PSO. The population size and the maximum number of iterations are set to 300 and 150, respectively. Table 3 shows the parameter settings for the considered algorithms. For a fair comparison among the algorithms, we have performed the Monte-Carlo simulations with 40 runs. The detailed comparison among the algorithms based on the performance indicators: solution quality, convergence speed, success and failure rate, and drone swarm formation running time are described as follows.

Figures 9a–c and 10a–c are the 2D and 3D views of the planned paths generated by PSO, IPSO, and ABC, respectively. The planned paths for the proposed iSCA are shown in Figs. 6b and 7b in 2D and 3D views, respectively. One can see from these figures that iSCA and IPSO can generate a feasible path for each drone, while PSO and ABC are not able to generate a feasible path. Moreover, ABC can not even maintain the safety distance among the drones as the collision happened among themselves also. This is because of the extensive exploration capability of ABC and unable to maintain the proper balance between exploration and exploitation. While due to the non-linearly decreasing step size and convergence factor in iSCA, it can easily construct a desired path for the drones while maintaining the safety distance among the drones. This comparison exhibits the advantages of the proposed method as compared to other considered algorithms.

In Tables 9 and 10, AFV of the formation and path failure rate (FR) of the considered algorithms are given, respectively. It is clear from Table 9 that iSCA obtains the least AFV with a reduced percentage of 4.1667, 1.9512, and 12.671 than PSO, IPSO, and ABC, respectively. This proves the ability of the proposed method to obtain the best solution than PSO, IPSO, and ABC. When the reliability of the algorithms is taken into consideration, Table 10 shows that iSCA has the highest reliability than that of PSO, IPSO, and ABC. In Table 10, one can see that the proposed iSCA has the least failure rate while ABC has the highest failure rate. The improved percentage in iSCA as compared to PSO, IPSO, and ABC are 60.416, 86.250, and 92.142, respectively. All these results demonstrate the effectiveness of the proposed iSCA both in terms of reliability as well as solution quality.

When the convergence speed is taken into account among the considered algorithms, Table 11 shows the average minimum iteration (AMI) of the formation that requires to satisfy the feasible criterion. The definition of feasible criterion is the same as defined above. Figure 11a–c represents the FV of PSO, IPSO, and ABC respectively, while Fig. 12a–c represents the AFV of PSO, IPSO, and ABC respectively. The AMI of the formation is graphically presented in Fig. 13. As it can be seen from Table 11, ABC has the least AMI, while IPSO is 2nd in the list. This shows that ABC has the highest convergence speed as compared to other algorithms. On the other hand, ABC has the largest function value. Therefore, though PSO, IPSO, and ABC have smaller AMI than iSCA, their function values are no longer outperformed iSCA. This is a clear indication of the fact that these algorithms suffers from premature convergence while iSCA continues to improve through iterations.

The formation running time for the considered algorithms is presented in Table 12 and Fig. 15b. It can be easily seen from Table 12 that the increased percentage of iSCA as compared to PSO, IPSO, and ABC are 1.8762, 20.2827, and 20.2827, respectively. Thus the proposed iSCA can generate feasible paths for the drone swarm more accurately and faster, as compared to PSO, IPSO, and ABC.

Conclusion and Future Direction

Drone swarm path planning problem in a 3D environment has been dealt with using an improved variant of the Sine Cosine Algorithm. A case study with 4 obstacles, 5 drones, and $16000 \times 16000 \times 16000$ size flying space has been used to check the performance of the proposed improved SCA (iSCA). The results are compared with the original SCA and other state-of-the-art meta-heuristic algorithms. The comparison results show that the proposed iSCA can generate the optimal paths for the drones more accurately with high convergence speed as compared to other considered algorithms. Our future research will concentrate on path-planning algorithms with dynamic obstacle environments. A further improvement of SCA or development of a new swarm intelligence-based algorithm for solving the drone swarm path planning problem in complex environments, such as re-planning the path in the event of unforeseen dangers or moving impediments will also be taken into account in future research. Furthermore, the development of an algorithm for re-planning the path of the swarm of drones when obstacles constantly change their shapes would also be an interesting and challenging future research topic.

Acknowledgements Author Jagdish Chand Bansal acknowledges the funding from Liverpool Hope University UK.

Author Contributions PP: Conceptualization, methodology, writing original draft. KP, AN: Review, JCB: Supervision, review.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Mozaffari M, Saad W, Bennis M, Debbah M. December. Drone small cells in the clouds: design, deployment and performance analysis. In: 2015 IEEE global communications conference (GLOBECOM). IEEE;2015. pp. 1–6.
2. Valavanis KP, Vachtsevanos GJ, editors. Handbook of unmanned aerial vehicles, vol. 1. Dordrecht: Springer; 2015.
3. Al-Hourani A, Kandeepan S, Lardner S. Optimal LAP altitude for maximum coverage. IEEE Wirel Commun Lett. 2014;3(6):569–72.
4. Gharibi M, Boutaba R, Waslander SL. Internet of drones. IEEE Access. 2016;4:1148–62.
5. Huo L, Zhu J, Wu G, Li Z. A novel simulated annealing based strategy for balanced UAV task assignment and path planning. Sensors. 2020;20(17):4769.
6. Ma Y, Hu M, Yan X. Multi-objective path planning for unmanned surface vehicle with currents effects. ISA Trans. 2018;75:137–56.
7. YongBo C, YueSong M, JianQiao Y, XiaoLong S, Nuo X. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. Neurocomputing. 2017;266:445–57.
8. Pehlivanoglu YV. A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. Aerosp Sci Technol. 2012;16(1):47–55.
9. Bayili S, Polat F. Limited-damage A*: a path search algorithm that considers damage as a feasibility criterion. Knowl Based Syst. 2011;24(4):501–12.
10. Baumann M, Leonard S, Croft EA, Little JJ. Path planning for improved visibility using a probabilistic road map. IEEE Trans Robot. 2010;26(1):195–200.
11. Brubaker MA, Geiger A, Urtasun R. Map-based probabilistic visual self-localization. IEEE Trans Pattern Anal Mach Intell. 2015;38(4):652–65.
12. Kothari M, Postlethwaite I. A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees. J Intell Robot Syst. 2013;71(2):231–53.
13. Moon CB, Chung W. Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree. IEEE Trans Ind Electron. 2014;62(2):1080–90.
14. Chen Y, Yu J, Su X, Luo G. Path planning for multi-UAV formation. J Intell Robot Syst. 2015;77(1):229–46.
15. Chen YB, Luo GC, Mei YS, Yu JQ, Su XL. UAV path planning using artificial potential field method updated by optimal control theory. Int J Syst Sci. 2016;47(6):1407–20.

16. Zhang X, Duan H. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. Appl Soft Comput. 2015;26:270–84.
17. Besada-Portas E, de la Torre L, Jesus M, de Andrés-Toro B. Evolutionary trajectory planner for multiple UAVs in realistic scenarios. IEEE Trans Robot. 2010;26(4):619–34.
18. Zheng C, Li L, Xu F, Sun F, Ding M. Evolutionary route planner for unmanned air vehicles. IEEE Trans Robot. 2005;21(4):609–20.
19. Ma Y, Zamirian M, Yang Y, Xu Y, Zhang J. Path planning for mobile objects in four-dimension based on particle swarm optimization method with penalty function. Math Probl Eng. 2013. https://doi.org/10.1155/2013/613964.
20. Ma Y, Hu M, Yan X. Multi-objective path planning for unmanned surface vehicle with currents effects. ISA Trans. 2018;75:137–56.
21. Ma H, Shen S, Yu M, Yang Z, Fei M, Zhou H. Multi-population techniques in nature inspired optimization algorithms: a comprehensive survey. Swarm Evol Comput. 2019;44:365–87.
22. Zhao Y, Zheng Z, Liu Y. Survey on computational-intelligence-based UAV path planning. Knowl Based Syst. 2018;158:54–64.
23. Roberge V, Tarbouchi M, Labonté G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Trans Ind Inf. 2012;9(1):132–41.
24. Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim. 2007;39(3):459–71.
25. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell Mag. 2006;1(4):28–39.
26. Konatowsłiowski P. Ant colony optimization algorithm for UAV path planning. In: 2018 14th International conference on advanced trends in radio electronics, telecommunications and computer engineering (TCSET). IEEE;2018. pp. 177–182.
27. Price KV. Differential evolution: a fast and simple numerical optimizer. In: Proceedings of North American fuzzy information processing. IEEE;1996. pp. 524–527.
28. Pan JS, Liu N, Chu SC. A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning. IEEE Access. 2020;8:17691–712.
29. James K, Russell E. Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol. 4. IEEE;1995.
30. Bansal JC, et al. Spider monkey optimization algorithm for numerical optimization. Memet Comput. 2014;6(1):31–47.
31. Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst. 2016;96:120–33.
32. Gupta S, Deep K. Improved sine cosine algorithm with crossover scheme for global optimization. Knowl Based Syst. 2019;165:374–406.
33. Nayak DR, et al. Combining extreme learning machine with modified sine cosine algorithm for detection of pathological brain. Comput Electr Eng. 2018;68:366–80.
34. Elaziz A, Mohamed DO, Xiong S. An improved opposition-based sine cosine algorithm for global optimization. Expert Syst Appl. 2017;90:484–500.
35. Duan H, Qiao P. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. Int J Intell Comput Cybern. 2014;7:24–37.
36. Wang G, Guo L, Duan H, Liu L, Wang H. A modified firefly algorithm for UCAV path planning. Int J Hybrid Inf Technol. 2012;5(3):123–44.
37. Zhu W, Duan H. Chaotic predator–prey biogeography-based optimization approach for UCAV path planning. Aerosp Sci Technol. 2014;32(1):153–61.
38. Ahmed G, Sheltami T, Mahmoud A, Yasar A. IoD swarms collision avoidance via improved particle swarm optimization. Transp Res Part A Policy Pract. 2020;142:260–78.
39. Weisstein EW. Hemisphere. 2002. https://mathworld.wolfram.com/.
40. Yang P, Tang K, Lozano JA, Cao X. Path planning for single unmanned aerial vehicle by separately evolving waypoints. IEEE Trans Robot. 2015;31(5):1130–46.
41. Gupta S. Enhanced sine cosine algorithm with crossover: a comparative study and empirical analysis. Expert Syst Appl. 2022;198:116856.
42. Shao S, Peng Y, He C, Du Y. Efficient path planning for UAV formation via comprehensively improved particle swarm optimization. ISA Trans. 2020;97:415–30.
43. Tian D, Shi Z. MPSO: modified particle swarm optimization and its applications. Swarm Evol Comput. 2018;41:49–68.