Safety-enhanced UAV Path Planning with Spherical Vector-based Particle Swarm Optimization

Manh Duong Phung[a,b,*], Quang Phuc Ha[a]

[a] *School of Electrical and Data Engineering, University of Technology Sydney (UTS)*
15 Broadway, Ultimo NSW 2007, Australia
[b] *VNU University of Engineering and Technology (VNU-UET), Vietnam National University, Hanoi (VNU)*
144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

Abstract

This paper presents a new algorithm named spherical vector-based particle swarm optimization (SPSO) to deal with the problem of path planning for unmanned aerial vehicles (UAVs) in complicated environments subjected to multiple threats. A cost function is first formulated to convert the path planning into an optimization problem that incorporates requirements and constraints for the feasible and safe operation of the UAV. SPSO is then used to find the optimal path that minimizes the cost function by efficiently searching the configuration space of the UAV via the correspondence between the particle position and the speed, turn angle and climb/dive angle of the UAV. To evaluate the performance of SPSO, eight benchmarking scenarios have been generated from real digital elevation model maps. The results show that the proposed SPSO outperforms not only other particle swarm optimization (PSO) variants including the classic PSO, phase angle-encoded PSO and quantum-behave PSO but also other state-of-the-art metaheuristic optimization algorithms including the genetic algorithm (GA), artificial bee colony (ABC), and differential evolution (DE) in most scenarios. In addition, experiments have been conducted to demonstrate the validity of the generated paths for real UAV operations. Source code of the algorithm can be found at `https://github.com/duongpm/SPSO`.

Keywords: Path planning, Particle swarm optimization, UAV

1. Introduction

Path planning is essential for UAVs to carry out tasks and avoid threats appearing in their operating environment. A planned path should be optimal in a specific criterion defined by the application. For most applications such as aerial photography, mapping, and surface inspection, the criterion is typically to minimize the traveling distance among the visiting locations of UAVs so that less time and fuel are required [1, 2]. The criterion can also be maximizing the detection probability as in dynamic target search [3], minimizing the flight time as with surveillance and rescue [4], or finding the Pareto solution for multi-objective navigation [5]. In addition, the planned path also needs to satisfy the constraints relating to safety imposed by the operating environment and feasibility imposed by the UAV. Here, safety relates to the capability of the path to guide the UAV through threats appearing in the environment such as obstacles. Feasibility involves the alignment of the path with UAV limits associated with flight time, flight altitude, fuel consumption, turning rate and climbing angle. Path planning with enhanced safety in terms of collision-free and feasible motion for UAVs therefore remains a challenging problem.

In the literature, several approaches have been proposed for UAV path planning such as graph search, cell decomposition, potential field and nature-inspired algorithms. The graph search approach splits the environment into connected discrete regions, each forms a vertex of the graph that the path is being searched. In [6, 7], the Voronoi diagram has been used to generate a graph which then became the input to the Eppstein's k-best paths algorithm [8] to find the best path. Another graph-based algorithm is the probabilistic roadmap (PRM) that samples the configuration space of the UAV to generate vertices of the graph [9]. Similar to PRM, the rapid-exploring random trees (RRT) algorithm uses the configuration space to create a search graph. It however finds the path by recursively adding the edge that has the smallest heuristic cost to it [10]. Although the graph-based algorithms are effective in generating feasible flight paths, they are not suitable to include constraints related to UAV maneuver and thus can result in large errors between the planned and flight paths.

The cell decomposition approach, on the other hand, represents the space as a grid of equal cells and employs a heuristic search to find the flight path. A* is a popular algorithm that searches the cell space using the least cost

*Corresponding author
Email addresses: `manhduong.phung@uts.edu.au` (Manh Duong Phung), `quang.ha@uts.edu.au` (Quang Phuc Ha)

from the current location to its neighbors and the target location [11, 12]. This algorithm is extended in [13] to include UAV constraints such as the turning angle. It is then modified to become bidirectional to deal with intermittent measurements [12]. Cell decomposition is also used in [14] for path coordination between UAVs and UGVs, in [15] for flight surveillance and in [16] for path prediction in real-time UAV operations. The main drawback of the cell decomposition approach however is the limitation in the scalable capacity as the number of cells exponentially increases with the search space dimension.

The potential field is another approach that directly searches the continuous space for solutions by treating the UAV as a particle moving under the influence of an artificial potential field constructed from components associated with the goal and any obstacles [17, 18]. This approach has been augmented with an additional control force to provide a shorter and smoother path [19, 20]. It is also combined with the Hamiltonian function to enable obstacle avoidance [21] or with the receding horizon optimization to obtain paths for multiple UAVs without violating the collision avoidance and network connectivity constraints [22]. The potential field approach, however, does not consider the optimality of the solution. It is also known to have limitations in dealing with local minima occurred in the field.

Recently, the nature-inspired approach has become more prevalent in path planning due to its effectiveness in dealing with UAV dynamic constraints and the capacity to search for the global optimum in complex scenarios. A variety of nature-inspired algorithms have been developed for UAV path planning such as the cuckoo search [23], genetic algorithm (GE) [24, 25], differential evolution (DE) [26, 27], artificial bee colony (ABC) [28], ant colony optimization (ACO) [29], and particle swarm optimization (PSO) [1, 25, 26, 30]. Among them, PSO is commonly used with a number of variants introduced.

Inspired by the behavior of bird flocking and fish schooling, PSO is a population-based algorithm that possesses two important properties of swarm intelligence, the cognitive and social coherence [31]. Those properties allow each particle of the swarm to search for the solution by following its own experience and the swarm experience instead of using conventional evolutionary operators like mutation and crossover. As a result, PSO is able to find the global solution with a stable convergence in a shorter computation time compared to other nature-inspired algorithms [32]. It is also known to be less sensitive to initial conditions and variations of objective functions and is able to adapt to various environment structures via a small number of parameters including one acceleration coefficient and two weight factors [33]. Due to its swarm nature, PSO can be parallelized to run on multiple processors, graphical processing units (GPU) or computer clusters to obtain the computation time required for both offline and online path planning [34]. Given those advantages, PSO has been widely used for path planning for mobile robot navigation with different approaches introduced such as evolutionary operator-based PSO [35], adaptive bare-bones PSO [36] or multi-objective PSO [37, 38]. In UAV path planning, several variants have been proposed such as the classic PSO [31, 25], phase angle-encoded PSO (θ-PSO) [39, 40, 30], quantum-behaved PSO (QPSO) [41, 30] and discrete PSO (DPSO) [1, 42]. Those variants have the same population-based structure but differ in the way they represent the search space and the solution encoded in particles. Consequently, different solutions may have resulted under the same conditions of the operating environment, dynamic constraints, and objective function. Therefore, it is important to compare those variants in different scenarios to provide a clear insight as to which of them is preferable for UAV path planning. In addition, it is also necessary to incorporate the maneuver properties of UAVs into the algorithms to further improve their navigation capacity.

In this study, we address the path planning problem for UAVs by first formulating an objective function that incorporates various requirements and constraints associated with the UAV and its flight path. We then introduce a new PSO algorithm that is capable of exploiting the configuration space of the UAV to generate quality solutions. For evaluation, eight scenarios have been generated with increasing levels of complexity based on the use of real digital elevation model (DEM) maps. The comparisons between SPSO and other PSO and metaheuristic algorithms are then conducted on those scenarios to evaluate their performance. In addition, experiments have been carried out to verify the feasibility of the solutions generated by SPSO for UAV operation in practical scenarios. Our contributions in this study therefore are fourfold: (i) development of a new objective function that converts the path planning into an optimization problem incorporating optimal criteria and constraints associated with the path length, threat, turn angle, climb/dive angle, and flight height for the safe and efficient operation of UAVs; (ii) proposal of a new PSO algorithm named spherical vector-based PSO (SPSO) that is capable of searching the configuration space for the global optimal solution; (iii) benchmarking the performance of PSO variants including PSO, θ-PSO, QPSO and SPSO for UAV path planning; (iv) validating the generated paths for real UAV operations.

The rest of this paper is structured as follows. Section 2 introduces the steps to formulate the objective function. Section 3 describes PSO and its variants. Section 4 presents SPSO and its implementation for solving the path planning problem. Section 5 provides comparison and experiment results. Finally, a conclusion is drawn to end our paper.

2. Problem Formulation

In this study, the path planning problem is formulated via a cost function that incorporates optimal criteria and UAV constraints described as follows.

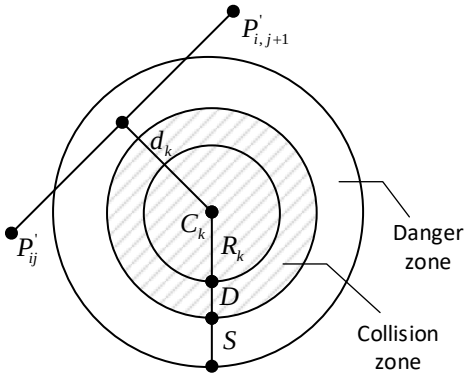


Figure 1: Determination of the threat cost.

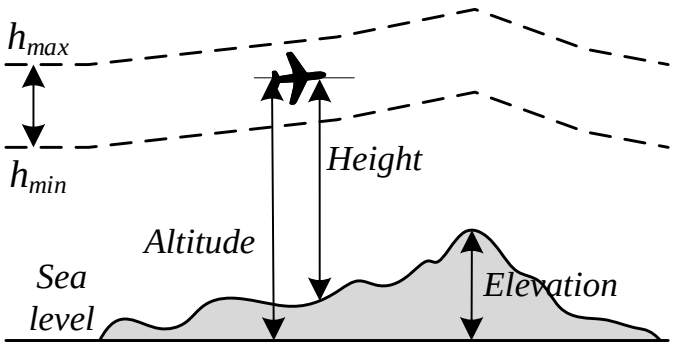


Figure 2: Altitude cost explanation.

2.1. Path optimality

For efficient operation of UAVs, a planned path needs to be optimal in a certain criterion depending on the application. With our focus on aerial photography, mapping, and surface inspection, we choose to minimize the path length. Since the UAV is controlled via a ground control station (GCS), a flight path X_i is represented as a list of n waypoints that the UAV needs to fly through. Each waypoint corresponds to a path node in the search map with coordinates $P_{ij} = (x_{ij}, y_{ij}, z_{ij})$. By denoting the Euclidean distance between two nodes as $\left\|\overrightarrow{P_{ij}P_{i,j+1}}\right\|$, the cost F_1 associated to the path length can be computed as:

$$F_1(X_i) = \sum_{j=1}^{n-1} \left\|\overrightarrow{P_{ij}P_{i,j+1}}\right\|. \tag{1}$$

2.2. Safety and feasibility constraints

Apart from optimality, the planned path needs to ensure the safe operation of the UAV by guiding it through threats that are typically caused by obstacles appearing in the operation space. Let K be the set of all threats, each is assumed to be prescribed in a cylinder with its projection having the center coordinate C_k and radius R_k as shown in Fig.1. For a given path segment $\left\|\overrightarrow{P_{ij}P_{i,j+1}}\right\|$, the associated threat cost is proportional to its distance, d_k, to C_k. By considering the diameter, D, of the UAV and the danger distance, S, to the collision zone, the threat cost F_2 is computed across waypoints P_{ij} for obstacle set K as follows:

$$\begin{cases} F_2(X_i) = \sum\limits_{j=1}^{n-1}\sum\limits_{k=1}^{K} T_k(\overrightarrow{P_{ij}P_{i,j+1}}), \\ T_k(\overrightarrow{P_{ij}P_{i,j+1}}) = \begin{cases} 0, & \text{if } d_k > S + D + R_k \\ (S + D + R_k) - d_k, & \text{if } D + R_k < d_k \le S + D + R_k \\ \infty, & \text{if } d_k \le D + R_k. \end{cases} \end{cases} \tag{2}$$

Note that while diameter D is determined by the UAV size, distance S depends on several factors such as the application, operating environment and positioning accuracy. For instance, S can be chosen from tens of meters in static environments with good GPS signal to hundreds of meters for environments with moving objects and weak GPS signal for positioning.

During operation, the flying altitude is often constrained between the two given extrema, the minimum and maximum heights. For example with surveying and search applications, it is required the visual data to be collected by the camera at a specific resolution and field of view and thus constrain the flying altitude. Let the minimum and maximum heights to be $h_{\min}$ and $h_{\max}$ respectively. The altitude cost associated to a waypoint P_{ij} is computed as:

$$H_{ij} = \begin{cases} |h_{ij} - \dfrac{(h_{\max} + h_{\min})}{2}|, & \text{if } h_{\min} \le h_{ij} \le h_{\max} \\ \infty, & \text{otherwise,} \end{cases} \tag{3}$$

where h_{ij} denotes the flight height with respect to the ground as illustrated in Fig.2. It can be seen that H_{ij} maintains the average height and penalises the out-of-range values. Summing H_{ij} for all waypoints gives the altitude cost:

$$F_3(X_i) = \sum_{j=1}^{n} H_{ij}. \tag{4}$$

The smooth cost evaluates the turning and climbing rates which are essential to generate feasible paths. As shown in Fig.3, the turning angle, ϕ_{ij}, is the angle between two consecutive path segments, $\overrightarrow{P'_{ij}P'_{i,j+1}}$ and $\overrightarrow{P'_{i,j+1}P'_{i,j+2}}$, projected on the horizontal plane Oxy. Let $\overrightarrow{k}$ be the unit vector in the direction of the z axis, the projected vector can be calculated as:

$$\overrightarrow{P'_{ij}P'_{i,j+1}} = \overrightarrow{k} \times (\overrightarrow{P_{ij}P_{i,j+1}} \times \overrightarrow{k}), \tag{5}$$

Hence, the turning angle is computed as:

$$\phi_{ij} = \arctan\left(\frac{\left\|\overrightarrow{P'_{ij}P'_{i,j+1}} \times \overrightarrow{P'_{i,j+1}P'_{i,j+2}}\right\|}{\overrightarrow{P'_{ij}P'_{i,j+1}}.\overrightarrow{P'_{i,j+1}P'_{i,j+2}}}\right). \tag{6}$$

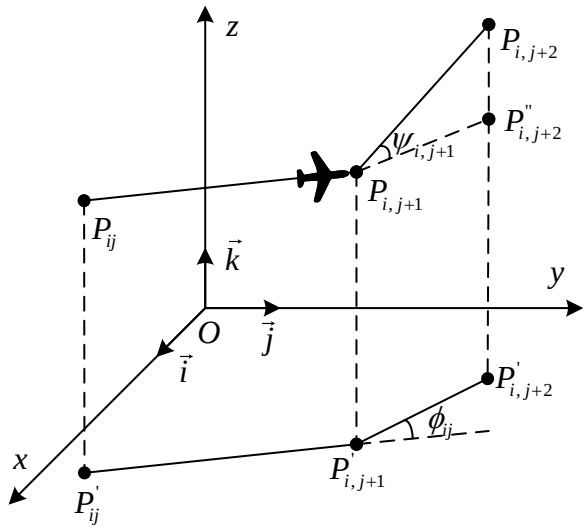


Figure 3: Turning and climbing angle calculation.

The climbing angle, ψ_{ij}, is the angle between the path segment $\overrightarrow{P_{ij}P_{i,j+1}}$ and its projection $\overrightarrow{P'_{ij}P'_{i,j+1}}$ onto the horizontal plane. It is given by:

$$\psi_{ij} = \arctan\left(\frac{z_{i,j+1} - z_{ij}}{\left\|\overrightarrow{P'_{ij}P'_{i,j+1}}\right\|}\right). \quad (7)$$

The smooth cost is then computed as:

$$F_4(X_i) = a_1 \sum_{j=1}^{n-2} \phi_{ij} + a_2 \sum_{j=1}^{n-1} \mid \psi_{ij} - \psi_{i,j-1} \mid, \quad (8)$$

where a_1 and a_2 are respectively the penalty coefficients of the turning and climbing angles.

2.3. Overall cost function

By considering the optimality, safety and feasibility constraints associated with a path X_i, the overall cost function can be defined of the form:

$$F(X_i) = \sum_{k=1}^{4} b_k F_k(X_i), \quad (9)$$

where b_k is the weight coefficient, and $F_1(X_i)$ to $F_4(X_i)$ are respectively the costs associated to the path length (1), threat (2), smoothness (4), and flight height (8). The decision variable is X_i including the list of n waypoints $P_{ij} = (x_{ij}, y_{ij}, z_{ij})$ such that $P_{ij} \in O$, where O is the operating space of UAVs. Given those definitions, the cost function F is fully determined and can be used as the input for the path planning process.

3. Related PSO algorithms for UAV path planning

With the cost function F defined in (9), the path planning becomes an optimization problem in which the aim is to find the path X^* that minimizes F. As F in general is a complicated multimodal function, solving it using classic methods such as hill climbing is not feasible due to local maxima. Instead, heuristic and metaheuristic methods are often used to provide quality solutions in a reasonable amount of time. This section describes the classic PSO and its variants including θ-PSO and QPSO which are among the most popular metaheuristic algorithms used for UAV path planning.

3.1. Particle swarm optimization

PSO is a stochastic optimization method working in light of swarm intelligence. Each particle i in the swarm is characterized by its position, $X_i = (x_{i1}, x_{i2}, ..., x_{iN})$, and velocity, $V_i = (v_{i1}, v_{i2}, ..., v_{iN})$, in the search space of N dimensions. It searches for the optimal solution by compromising between its own experience reflected via the local best position, $L_i = (l_{i1}, l_{i2}, ..., l_{iN})$, and the swarm experience reflected via the global best position, $L_g = (L_{g1}, L_{g2}, ..., L_{gN})$. For a swarm of M particles, the compromise is carried out by the following equations:

$$v_{ij}^{k+1} \leftarrow w^k v_{ij}^k + \eta_1 r_{1j}(l_{ij}^k - x_{ij}^k) + \eta_2 r_{2j}(l_{gj}^k - x_{ij}^k) \quad (10)$$

$$x_{ij}^{k+1} \leftarrow x_{ij}^k + v_{ij}^{k+1}, (i = 1, 2, ..., M; j = 1, 2, ..., N), \quad (11)$$

where k represents the kth generation, $x_{ij} \in [x_{min}, x_{max}]$ and $v_{ij} \in [v_{min}, v_{max}]$ are respectively the jth dimension of the ith particle's position and velocity, w^k is the inertial weight, η_1 and η_2 are respectively the cognitive and social coefficients, and r_{1j} and r_{2j} are two random samples within $[0, 1]$ drawn from a uniform probability distribution. The values of η_1 and η_2 determine the moving tendency of particles toward the local best and global best position. The weight w^k, on the other hand, represents the compromise between the exploration (global search) and exploitation (local search). It is often chosen to be smaller over generations to increase the exploitation when the swarm is getting closer to the optimal solution.

When using PSO for UAV path planning, the position of each particle encodes a candidate path. Hence, the swarm is equivalent to a matrix of M paths, $X = [X_1, X_2, ..., X_M]^T$, each includes a list of N waypoints of the form:

$$X_i = (x_{i1}, y_{i1}, z_{i1}, x_{i2}, y_{i2}, z_{i2}, ..., x_{i,N}, y_{i,N}, z_{i,N}). \quad (12)$$

As the start and end points of all paths are fixed, they are not included in the particle position. A path of n waypoints is thus represented by a particle of dimension $3N$, $N = n - 2$. During the optimization process, the particles evolve according to (15) and (16) based on the evaluation in (9) to converge to the best path.

3.2. Phase angle-encoded particle swarm optimization

θ-PSO uses angles instead of Cartesian coordinates to represent particle positions [39, 40, 30]. In θ-PSO, a path

of n waypoints is described by a vector of $3N$ angles:

$$\Theta_i = (\theta_{i1}, ..., \theta_{iN}, \theta_{i,N+1}, ..., \theta_{i,2N}, \theta_{i,2N+1}, ..., \theta_{i,3N}), \tag{13}$$

where $N = n-2$ and θ_{ij} is within the interval $[-\pi/2, \pi/2]$. The velocity associated to each particle is then expressed by angle increments as:

$$\Delta\Theta_i = (\Delta\theta_{i1}, ..., \Delta\theta_{iN}, \Delta\theta_{i,N+1}, ..., \Delta\theta_{i,2N}, \Delta\theta_{i,2N+1}, ..., \Delta\theta_{i,3N}). \tag{14}$$

Hence, the update equations for θ-PSO are given by:

$$\Delta\theta_{ij}^{k+1} \leftarrow w^k \Delta\theta_{ij}^k + \eta_1 r_{1j}(\gamma_{ij}^k - \theta_{ij}^k) + \eta_2 r_{2j}(\gamma_{gj}^k - \theta_{ij}^k) \tag{15}$$

$$\theta_{ij}^{k+1} \leftarrow \theta_{ij}^k + \Delta\theta_{ij}^{k+1}, (i = 1, 2, ..., M; j = 1, 2, ..., 3N), \tag{16}$$

where $\Gamma_i = [\gamma_{i1}, \gamma_{i2}, ..., \gamma_{i,3N}]$ and $\Gamma_g = [\gamma_{g1}, \gamma_{g2}, ..., \gamma_{g,3N}]$ are respectively the phase angles of the local and global best positions of particle i.

To evaluate the fitness, a monotonic function is used to map particles from the angular space to the coordinate space. Let the monotonic function be $f : [-\pi/2, \pi/2] \rightarrow [x_{min}, x_{max}]$, there is one and only one position X_i mapped by f corresponding to any given position Θ_i. That position is given by [39]:

$$\begin{cases} x_{ij} = f(\theta_{ij}), \\ f(\theta_{ij}) = \dfrac{1}{2}[(x_{max} - x_{min})sin(\theta_{ij}) + x_{max} + x_{min}]. \end{cases} \tag{17}$$

It can be seen from (17) that θ-PSO introduces nonlinearity to the candidate paths by adding more waypoints to their middle section and thus aims to improve the search capacity in that area of the operating environment.

3.3. Quantum-behaved particle swarm optimization

QPSO assumes particles have a quantum state described by a wavefunction $\psi(x, t)$ and is attracted by a Delta potential well [41]. The probability that a particle appears at position x is then described via its probability density function $|\psi(x, t)|^2$, which can be derived from the time-dependent Schrödinger equation. By using the Monte Carlo simulation method, that position is updated by the following equation:

$$x_{ij}^{k+1} = p_{ij}^k \pm 0.5 L_{ij}^k ln(1/r), \tag{18}$$

where $r \in (0, 1)$ is a random number drawn from a uniform probability distribution and p_{ij}^k is a local attractor computed as:

$$p_{ij}^k = a l_{ij}^k + (1 - a) l_{gj}^k, \tag{19}$$

where $a \in (0, 1)$ is a random number of uniform distribution. L_{ij}^k is the parameter computed by:

$$L_{ij}^k = 2\beta |mbest_j^k - x_{ij}^k|, \tag{20}$$

$$mbest_j^k = \sum_{i=1}^{M} l_{ij}^k / M, \tag{21}$$

where *mbest* is the mean best position of the swarm and β is the contraction-expansion coefficient. Noting that particles in QPSO do not maintain the velocity component but only the position.

In path planning, QPSO represents a flight path as a set of N waypoints similar to PSO. Those waypoints are encoded through the position of particles which is then updated by 18.

4. Spherical vector-based PSO for UAV path planning

Exploiting maneuver characteristics of UAVs, we propose in this study the SPSO algorithm and provide its implementation to solve the path planning problem.

4.1. Spherical vector-based PSO algorithm

SPSO encodes each path as a set of vectors, each describes the movement of the UAV from one waypoint to another. Those vectors are represented in the spherical coordinate system with three components including the magnitude $\rho \in (0, path\_length)$, elevation angle $\psi \in (-\pi/2, \pi/2)$, and azimuth angle $\phi \in (-\pi, \pi)$. A flight path Ω_i with N nodes is then represented by a hyper spherical vector of $3N$ dimensions:

$$\Omega_i = (\rho_{i1}, \psi_{i1}, \phi_{i1}, \rho_{i2}, \psi_{i2}, \phi_{i2}, ..., \rho_{iN}, \psi_{iN}, \phi_{iN}), N = n-2. \tag{22}$$

By describing the position of a particle as Ω_i, the velocity associated to that particle is described by an incremental vector:

$$\Delta\Omega_i = (\Delta\rho_{i1}, \Delta\psi_{i1}, \Delta\phi_{i1}, \Delta\rho_{i2}, \Delta\psi_{i2}, \Delta\phi_{i2}, ..., \Delta\rho_{iN}, \Delta\psi_{iN}, \Delta\phi_{iN}) \tag{23}$$

Denoting spherical vector $(\rho_{ij}, \psi_{ij}, \phi_{ij})$ as u_{ij} and velocity $(\Delta\rho_{ij}, \Delta\psi_{ij}, \Delta\phi_{ij})$ as Δu_{ij}, the update equations for SPSO are given by:

$$\Delta u_{ij}^{k+1} \leftarrow w^k \Delta u_{ij}^k + \eta_1 r_{1j}(q_{ij}^k - u_{ij}^k) + \eta_2 r_{2j}(q_{gj}^k - u_{ij}^k) \tag{24}$$

$$u_{ij}^{k+1} \leftarrow u_{ij}^k + \Delta u_{ij}^{k+1}, (i = 1, 2, ..., M; j = 1, 2, ..., N), \tag{25}$$

where $Q_i = (q_{i1}, q_{i2}, ..., q_{i,N})$ and $Q_g = (q_{g1}, q_{g2}, ..., q_{g,N})$ are respectively the sets of vectors representing the local and global best positions of particle i.

In order to determine Q_i and Q_g, it is required to map a vector-based flight path Ω_i to a direct path X_i so that the associated cost can be evaluated. The mapping of vector $u_{ij} = (\rho_{ij}, \psi_{ij}, \phi_{ij}) \in \Omega_i$ to waypoint $P_{ij} = (x_{ij}, y_{ij}, z_{ij}) \in X_i$ can be conducted as:

$$x_{ij} = x_{i,j-1} + \rho_{ij} sin\psi_{ij} cos\phi_{ij}, \tag{26}$$

$$y_{ij} = y_{i,j-1} + \rho_{ij} sin\psi_{ij} sin\phi_{ij}, \tag{27}$$

$$z_{ij} = z_{i,j-1} + \rho_{ij} cos\psi_{ij}. \tag{28}$$

Denoting the map as $\xi : \Omega \rightarrow X$, the local and global best positions can be computed as:

$$Q_i = \begin{cases} \Omega_i & \text{if } F(\xi(\Omega_i)) < F(\xi(Q_{i-1})) \\ Q_{i-1} & \text{otherwise} \end{cases}, \tag{29}$$

$$Q_g = \underset{Q_i}{\operatorname{argmin}} F(\xi(Q_i)). \quad (30)$$

The rationale for the use of spherical vectors in SPSO is to achieve safety enhancement of navigation via the interrelationship between the magnitude, elevation and azimuth components of the vectors with the speed, turning angle and climbing angle of the UAV. As a result, the particles of SPSO search for solutions in the configuration space instead of the Cartesian space to reach a higher probability of finding quality solutions. More importantly, constraints relating to the turning and climbing angles can be directly implemented via the elevation and azimuth angles of the spherical vector so that the search space can be significantly reduced. In some scenarios, for example when the UAV flies at a constant speed, the magnitude can be fixed to further reduce the search space for extending the search capacity.

```
/* Initialization:                                   */
Get search map and initial path planning
 information ;
Set swarm parameters w, η1, η2, swarm_size;
foreach particle i in swarm do
    Create a random path Ω_i^0;
    Assign Ω_i^0 to particle's position;
    Compute fitness F(ξ(Ω_i)) of the particle;
    Set local_best Q_i of the particle to its fitness;
end
Set global_best Q_g to the best fit particle;
/* Evolutions:                                       */
for k ← 1 to max_generation do
    foreach particle i in swarm do
        Compute velocity ΔΩ_i^k;        /* Eq.24 */
        Compute new position Ω_i^k;     /* Eq.25 */
        Map Ω_i^k to X_i^k in Cartesian space; /* Eq.26
         - 28 */
        Update fitness F(X_i^k);         /* Eq.9 */
        Update local_best Q_i;          /* Eq.29 */
    end
    Update global_best Q_g;             /* Eq.30 */
    Save best_position Ω* associated with Q_g ;
     /* the best path */
end
```

Algorithm 1: Pseudo code of SPSO for UAV path planning.

4.2. Implementation of SPSO for UAV path planning

The pseudo code of SPSO is shown in Fig.1. It shares the same structure as other PSO variants including parameter initialization, particle generation and swarm evolution, but differs from others in the representation of particles' position, velocity and update equations. Therefore, parallelism can be used as adopted in [1] to speed up the calculation process. During algorithm execution, infeasible solutions appeared will be assigned an infinite cost value so that they will excluded from the final output solutions.

5. Results

To evaluate the performance of SPSO, we have conducted a number of comparisons and experiments with details as follows.

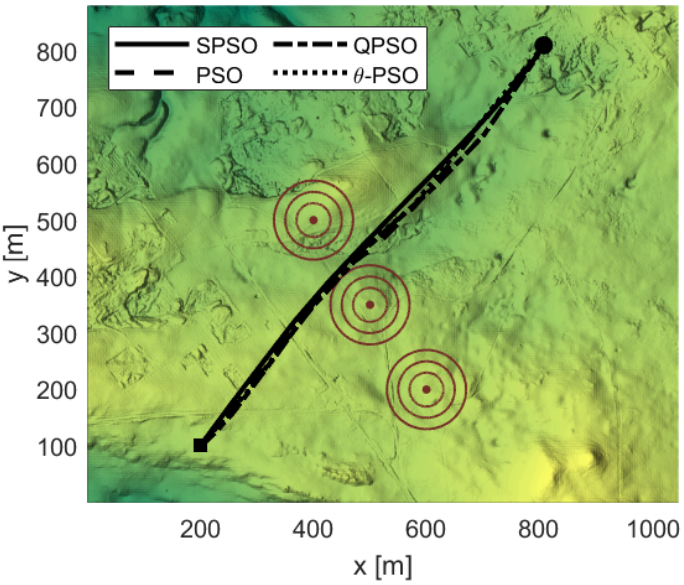
5.1. Scenario setup

The scenarios used for evaluation are based on real digital elevation model (DEM) maps derived from LiDAR sensors [43]. Two areas of Christmas Island in Australia with different terrain structures are selected and then augmented to generate eight benchmarking scenarios as shown in Fig.4 and Fig. 5. In those scenarios, the number and location of threats, represented as red cylinders, are chosen at different levels of complexity.

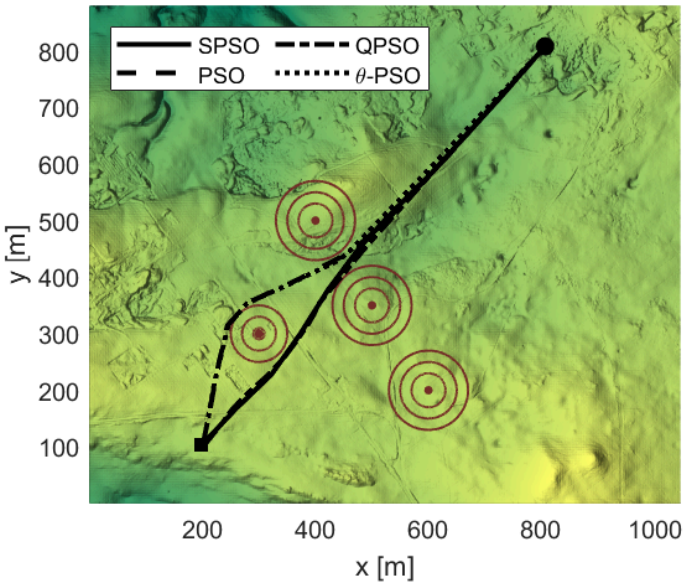
For comparisons, all PSO variants are implemented with the same set of parameters: $w = 1$ with the damping rate of 0.98, $\eta_1 = 1.5$ and $\eta_2 = 1.5$. The swarm size is chosen to be 500 particles and the number of iterations is 200. The number of waypoints are respectively selected as $n = 12$ and $n = 22$ corresponding to 10 and 20 line segments. In each comparison, all algorithms are run 10 times to find the average and standard deviation values. In addition, a statistical metric named paired sample t-test [44] is used to evaluate the significance of mean differences between SPSO and other PSO algorithms. The notation $D+$ implies that the mean value of SPSO is statistically better than the compared PSO, $D-$ implies the opposite, whereas N means that the difference is insignificant and NA stands for "Not Applicable". The level of confidence in t-test evaluations is set to $\alpha = 0.05$ equivalent to 95%.

5.2. Comparison between PSO algorithms

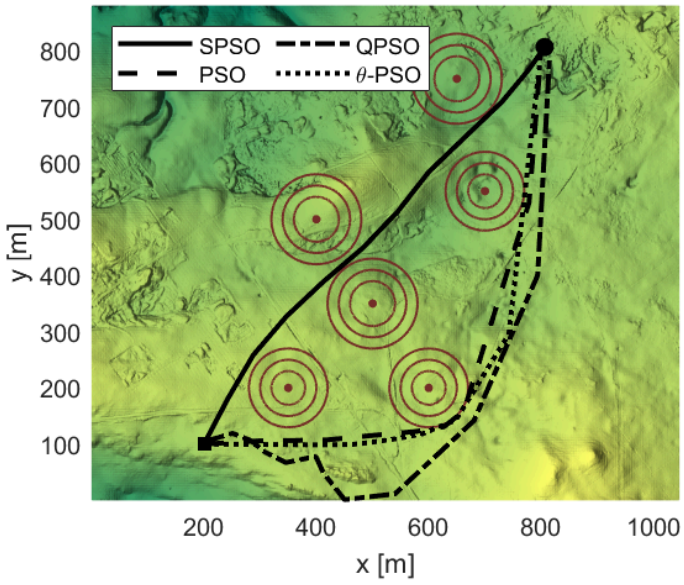
The top view of the resultant paths for $n = 12$ generated by PSO algorithms are shown in Fig.4 and Fig.5. It can be seen that all algorithms are able to generate feasible paths that fulfill the requirements on the path length, threat, turn angle, climb/dive angle and height. Their optimality, however, varies with scenarios. For simple scenarios 1, 2, 5, and 6, all algorithms converge well with slight differences in their fitness values. The t-test values in Table 1 also show that some differences in scenarios 1 and 5 are statistically insignificant. For more complicated scenarios 3, 4, 7, and 8, their performance however is much different. SPSO is able to obtain near-optimal solutions, whereas PSO and θ-PSO only converge to relatively good solutions. QPSO is not able to find quality solutions. This result can be further confirmed by Table 1 which presents the average, standard deviation, and paired sample t-test of the fitness values. It shows that SPSO statistically achieves the best fitness with $D+$ t-test in most scenarios while QPSO is only good in simple scenarios. PSO and θ-PSO introduce



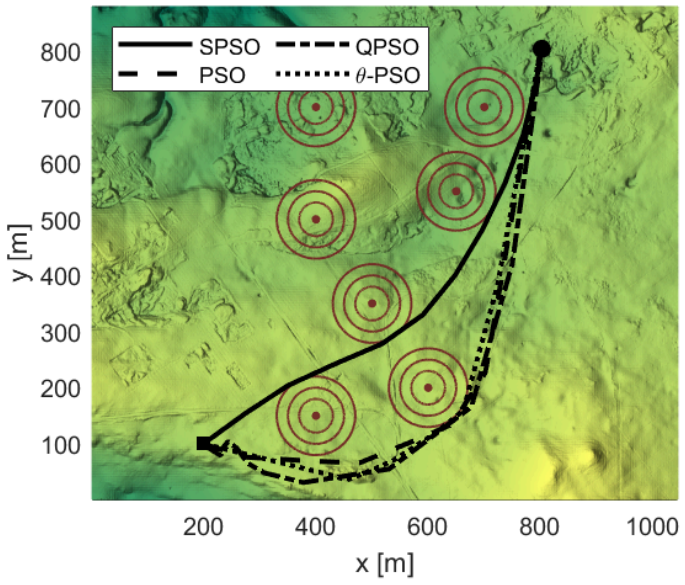
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

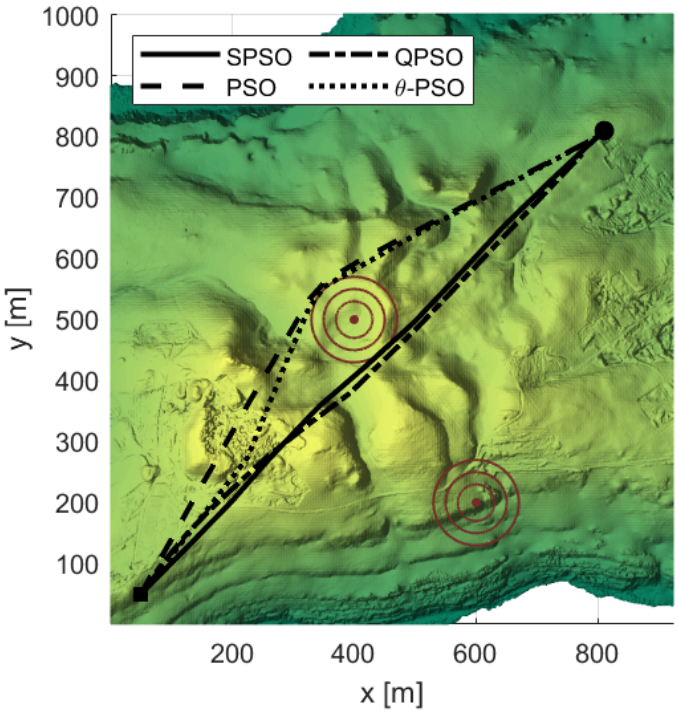


(d) Scenario 4

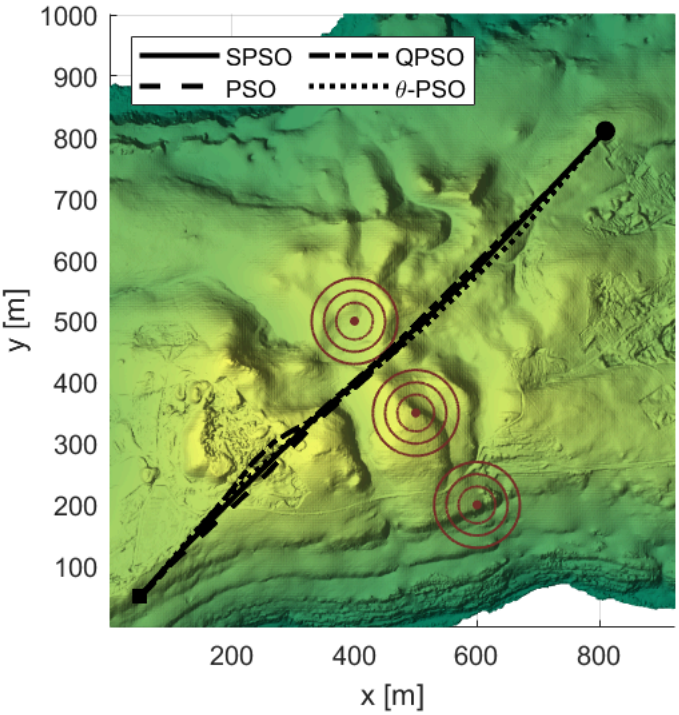
Figure 4: Top view of the paths generated by the PSO variants for scenarios 1 to 4

Table 1: Fitness values of the paths generated by the PSO variants with 10 line segments ($n = 12$)

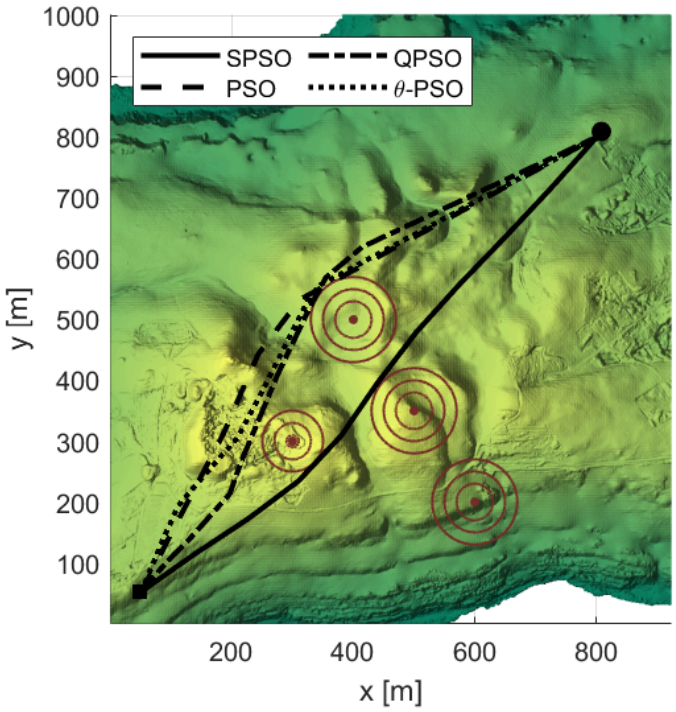
Scenario	SPSO			PSO			θ-PSO			QPSO		
-	Mean	Std	t-test	Mean	Std	t-test	Mean	Std	t-test	Mean	Std	t-test
1	4683	104	NA	4683	98	N	**4643**	50	N	4826	162	$D+$
2	**4699**	94	NA	5059	41	$D+$	5006	69	$D+$	5958	220	$D+$
3	**5486**	38	NA	5761	20	$D+$	5766	32	$D+$	7470	462	$D+$
4	**4994**	28	NA	5781	56	$D+$	5794	46	$D+$	7120	761	$D+$
5	**5441**	27	NA	5476	37	N	5518	37	$D+$	5508	33	$D+$
6	**5362**	59	NA	5514	67	$D+$	5486	45	$D+$	5474	21	$D+$
7	**5778**	94	NA	5838	39	$D+$	5800	43	N	5965	193	$D+$
8	**6006**	63	NA	6396	29	$D+$	6368	46	$D+$	8093	259	$D+$



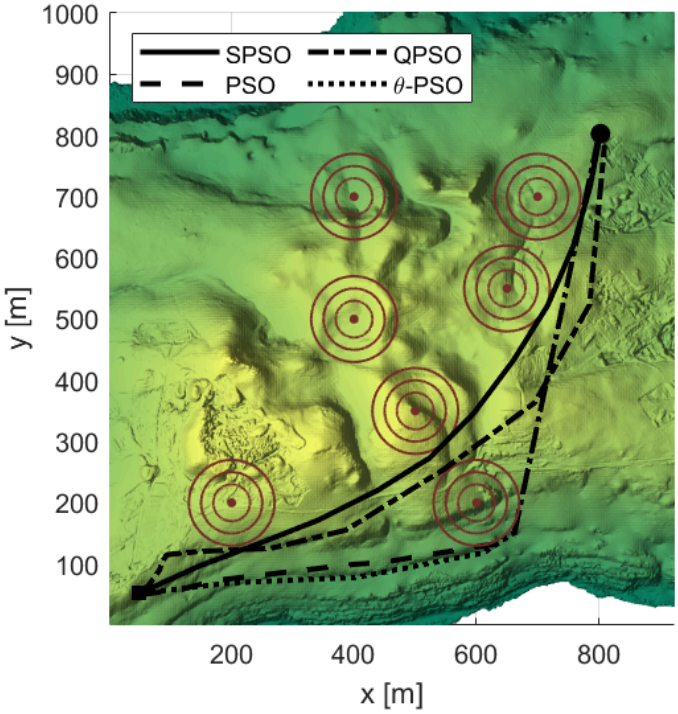
(a) Scenario 5



(b) Scenario 6



(c) Scenario 7



(d) Scenario 8

Figure 5: Top view of the paths generated by the PSO variants for scenarios 5 to 8

relatively good results in all scenarios with stable convergence reflected via small deviations.

Figure 6 provides a closer look at the behavior of the variants by showing their best fitness over iterations. It is recognizable that all variants converge in a similar fashion except QPSO. It is due to the fact that QPSO does not originate from the interaction of biological swarms but the transition in quantum states of particles. On another note, SPSO presents the best performance as it has a direct mapping between the properties of particles and UAV parameters to gain advantages in exploring the search space. Figure 7 shows the 3D and side views of the paths obtained by SPSO for scenarios 4 and 8, the two most challenging scenarios. It can be seen that the paths are smooth and valid with the flight height maintained properly with respect to the terrain.

To compare the scalability of PSO variants, we increased the number of waypoints to $n = 22$ in another comparison. The result presented in Table 2 shows that QPSO does not perform well in most scenarios, especially scenarios 3, 4 and 8 when its particles could not evolve to find better solutions. PSO and θ-PSO perform properly for simple scenarios but for complicated scenarios like 4, 5, and 8, the quality of solutions is degraded due to their limitation in exploring large search space. SPSO, on the other hand, performs well in most scenarios thanks to the spherical vector-based encoding mechanism that allows its particles to search in the configuration space.

5.3. Comparison with other metaheuristic algorithms

To further evaluate the performance of SPSO, we have compared its performance with other state-of-the-art metaheuristic algorithms including the genetic algorithm (GA), artificial bee colony (ABC), and differential evolution (DE). GA is implemented as in [25] with three mutation operations: add a node, delete a node and merge two nodes. ABC is implemented in its standard form [45]. DE is also implemented in its standard form [46], but with some changes in the swarm size and iterations due to its characteristic which performs better over a large number of iterations [47]. Specifically, DE is implemented with the swarm size of 100 and the iteration number of 1000 to ensure the algorithm convergence at the same number of fitness evaluations as with other algorithms.

Table 3 shows the fitness results. It can be seen that SPSO outperforms other algorithms with increasing margins and $D+$ t-test for complicated scenarios 3, 4, 7 and 8. However, DE performs well for complicated scenarios and has a stable convergence with small deviations due to its exploration capacity over a large number of iterations. ABC performs relatively well while GA shows the least stable performance. The reason for the unstable performance of GA lies in its operator 'delete a node'. This operator removes waypoints from a path to reduce the search dimensions so that the probability of finding quality solutions is increased. That operator, however, also reduces the resolution of candidate paths causing them insufficient to adapt to complex threats as with scenarios 3, 4, and 8.

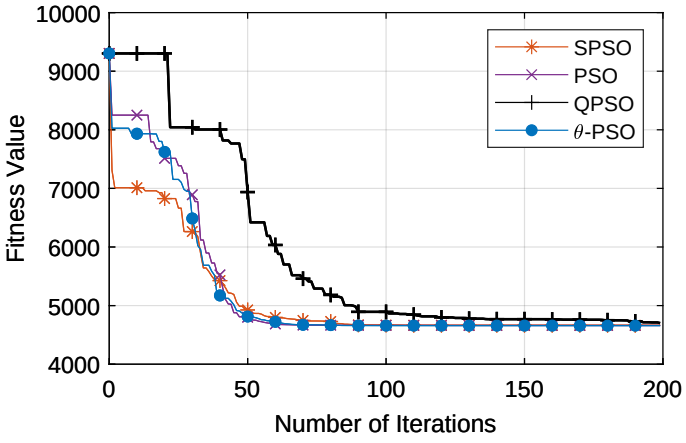
Figures 8 and 9 provide the top view of the paths generated. It can be seen that all algorithms are able to generate collision free paths. However, SPSO introduces the smoothest and shortest paths in most scenarios. DE does not introduce the best paths for simple scenarios but provides near-optimal solutions for the complicated ones. It reflects the nature of DE that carries out exploration via mutation and selection to generate quality solutions but is limited in exploitation to find the optimal solutions. Similarly, ABC tends to generate average paths due to the compromise among different types of bees. Finally, GA generates paths that consist of only several line segments and sharp turns as the result of node deletion.

Figure 10 shows the best fitness over iterations where the values obtained by DE is scaled to 200 iterations for the sake of comparison. It can be seen that GA converges quickly to premature solutions due to search dimension reduction. ABC presents slow convergence because of its weakness in exploration. DE has steady convergence which shows the efficiency of its differential operator. Finally, SPSO has sufficiently fast convergence due to the balance between exploitation and exploration implemented via the social and coherence coefficients.

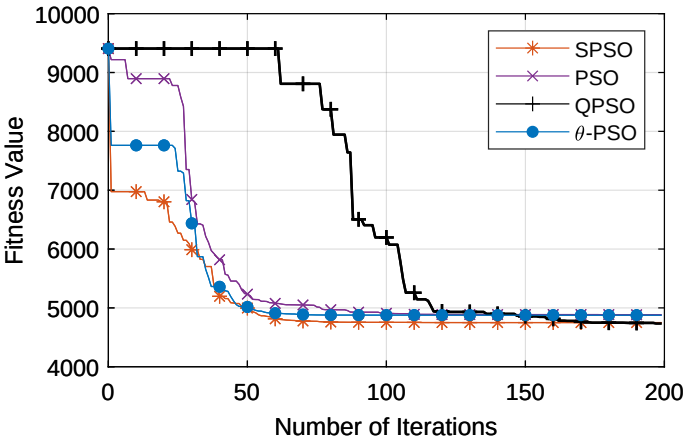
5.4. Experimental verification

We have conducted several experiments to evaluate the validity of the generated paths for real UAV operations. The UAV used is a 3DR Solo drone that can be programmed to fly automatically via ground control station software named Mission Planner as shown in Fig. 11a. The field used is a park in Sydney which has a monorail bridge that the drone needs to flight through as shown in Fig. 11b. The field is augmented with threats to create two experimental scenarios. Scenario 1 has a flat surface with 5 threats, whereas scenario 2 has 4 threats and includes the monorail bridge with sharp changes in height as shown in Fig. 12a and Fig. 12b. The longitude and latitude of the start and goal locations for scenario 1 is (-33.87643,151.191778) and (-33.875711,151.192643), and scenario 2 is (-33.875849, 151.191528) and (-33.87513,151.192394) respectively. Those locations, together with the terrain map and flight constraints are used as inputs of SPSO to generate waypoints. The waypoints are then uploaded to the drone via Mission Planner for autonomous flight.

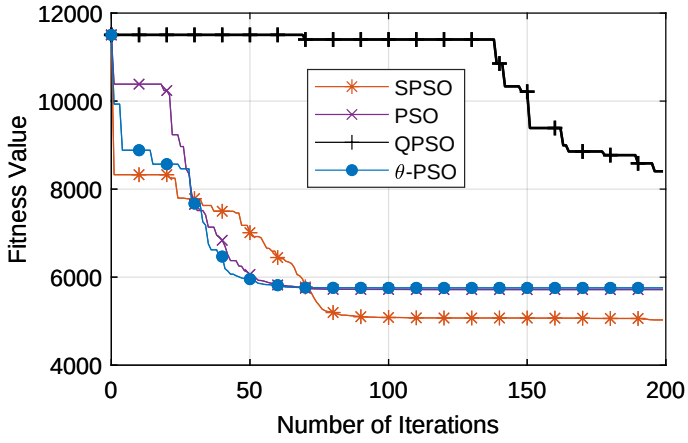
Figure 12a shows the planned and flight paths obtained in real time from Mission Planner for scenario 1. It can be seen that the flight path is collision free and overlaps well with the planned path. The flight height, which is basically constant, also matches the planned path as shown in Fig. 12c. Similar results have been obtained for scenario 2 as shown in Fig. 12b and Fig. 12d. Notably, the drone can track the planned path to carry out abrupt changes in height to fly over the monorail bridge. Besides, the good match between the planned and flight paths indicates not



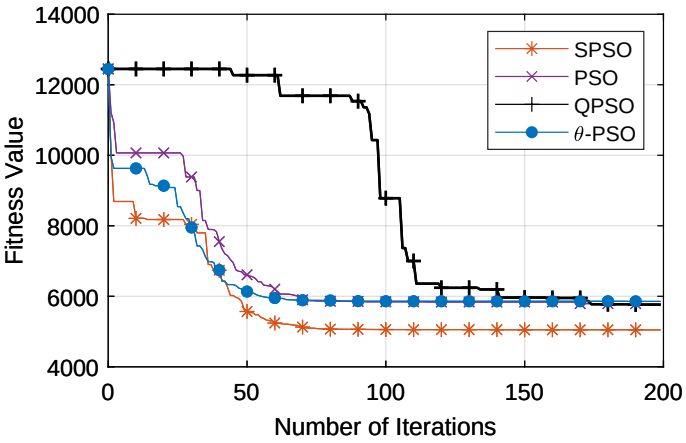
(a) Scenario 1



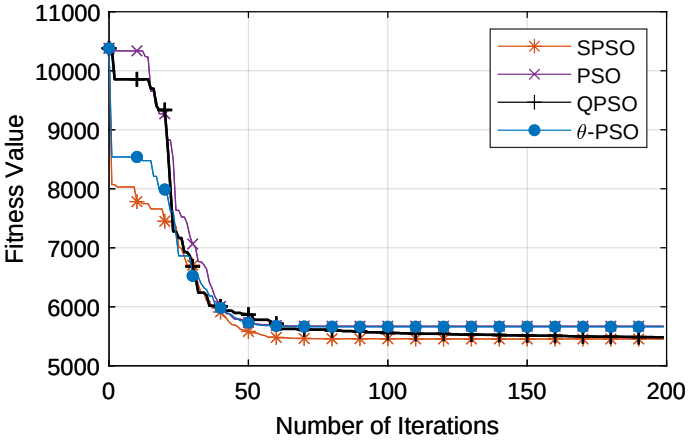
(b) Scenario 2



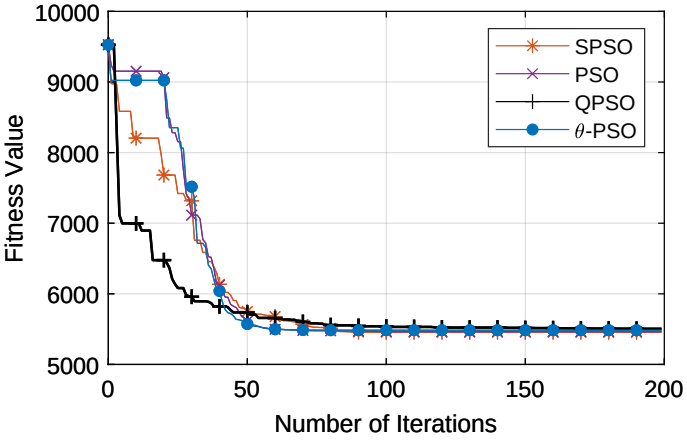
(c) Scenario 3



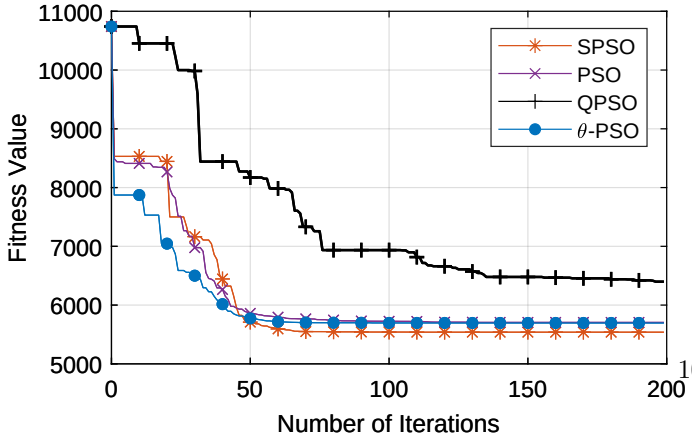
(d) Scenario 4



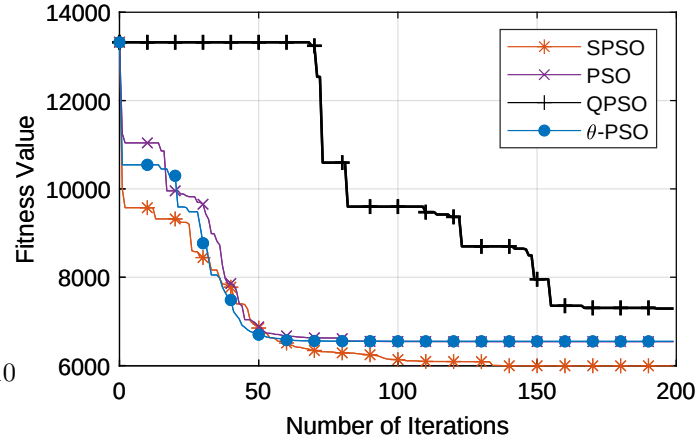
(e) Scenario 5



(f) Scenario 6

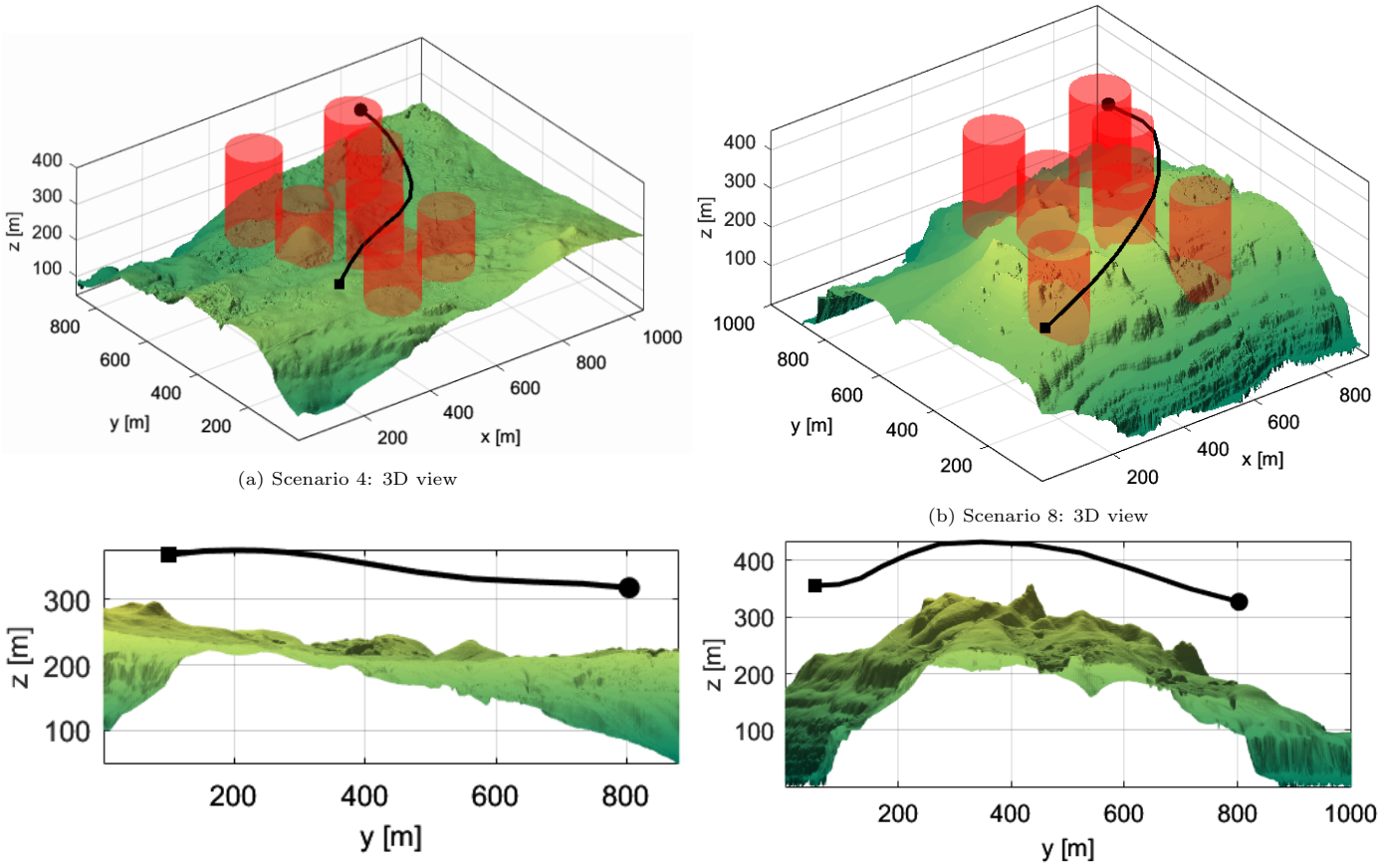


(g) Scenario 7



(h) Scenario 8

Figure 6: Best fitness values over iterations of the PSO algorithms



(a) Scenario 4: 3D view

(b) Scenario 8: 3D view

(c) Scenario 4: Side view

(d) Scenario 8: Side view

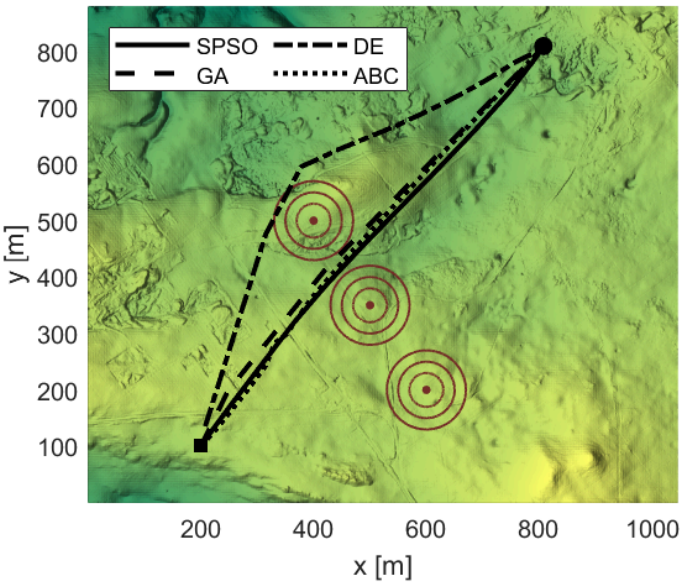
Figure 7: The planned paths generated by SPSO for scenarios 4 and 8

Table 2: Fitness values of the paths generated by the PSO variants with 20 line segments ($n = 22$)

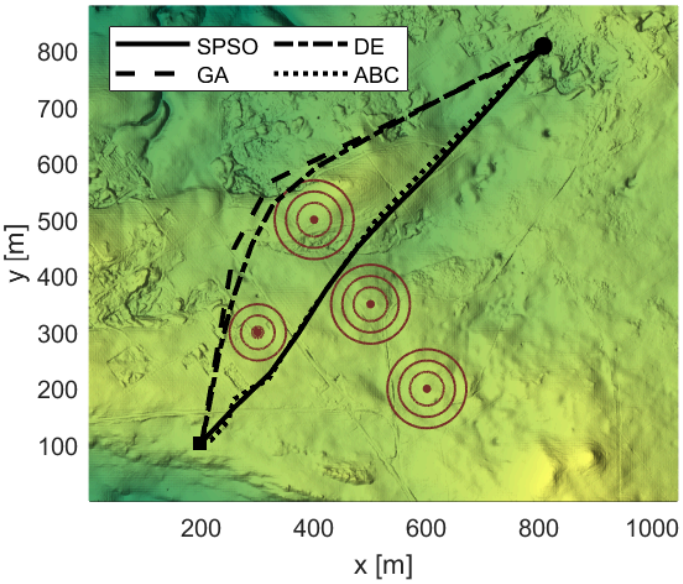
Scenario	SPSO			PSO			θ-PSO			QPSO		
-	Mean	Std	t-test	Mean	Std	t-test	Mean	Std	t-test	Mean	Std	t-test
1	**4757**	91	NA	4821	54	N	4827	63	N	5246	236	$D+$
2	4906	141	NA	**4900**	162	N	4903	136	N	5379	201	$D+$
3	**6202**	172	NA	6521	291	$D+$	6260	355	N	16926	0	$D+$
4	**5373**	179	NA	6146	335	$D+$	6201	451	$D+$	15406	354	$D+$
5	**5806**	222	NA	6188	133	$D+$	6330	237	$D+$	6572	320	$D+$
6	**5718**	130	NA	5844	275	$D+$	5733	173	N	6049	68	$D+$
7	**5951**	132	NA	5992	197	N	6338	363	$D+$	7143	372	$D+$
8	**6152**	111	NA	7016	281	$D+$	7134	782	$D+$	13828	0	$D+$

Table 3: Fitness values of the paths generated by the SPSO and other metaheuristic algorithms with 10 line segments ($n = 12$)

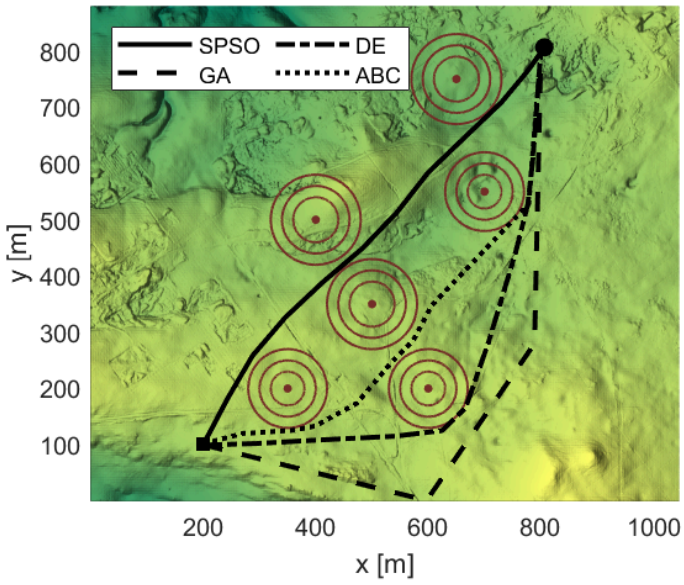
Scenario	SPSO			GA			DE			ABC		
-	Mean	Std	t-test	Mean	Std	t-test	Mean	Std	t-test	Mean	Std	t-test
1	**4683**	104	NA	4782	145	N	5014	6	$D+$	4822	49	$D+$
2	**4699**	94	NA	5357	113	$D+$	5040	14	$D+$	5020	56	$D+$
3	**5486**	38	NA	6761	94	$D+$	5716	2	$D+$	5882	266	$D+$
4	**4994**	28	NA	6325	224	$D+$	5741	4	$D+$	5325	118	$D+$
5	**5441**	27	NA	5676	117	$D+$	5482	9	$D+$	5608	34	$D+$
6	**5362**	59	NA	5424	81	$D+$	5665	36	$D+$	5676	41	$D+$
7	5778	94	NA	5919	75	$D+$	**5633**	17	$D-$	5976	75	$D+$
8	**6006**	63	NA	7274	554	$D+$	6290	72	$D+$	6719	90	$D+$



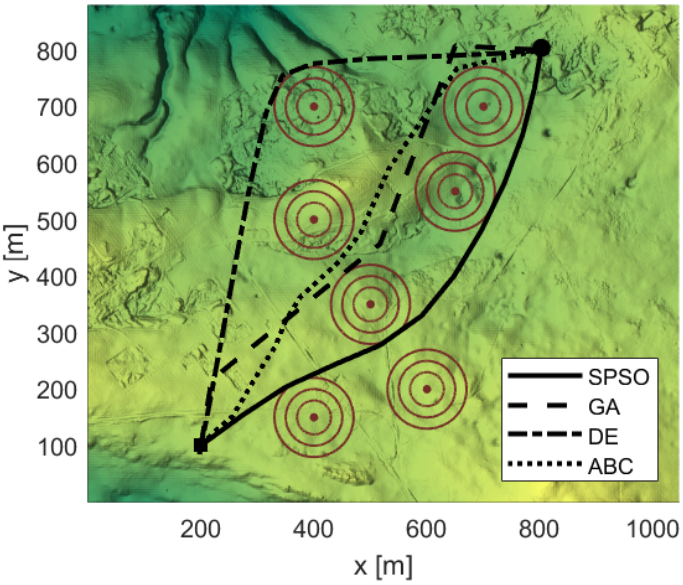
(a) Scenario 1



(b) Scenario 2

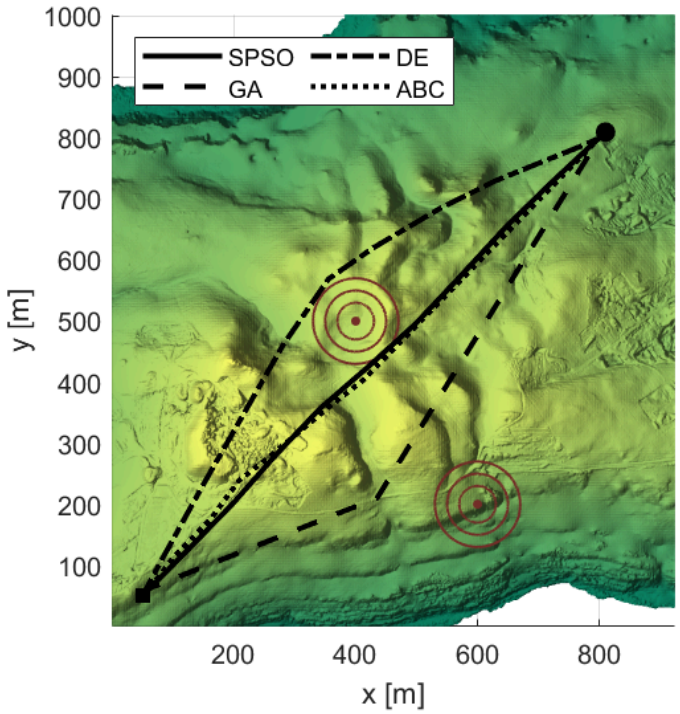


(c) Scenario 3

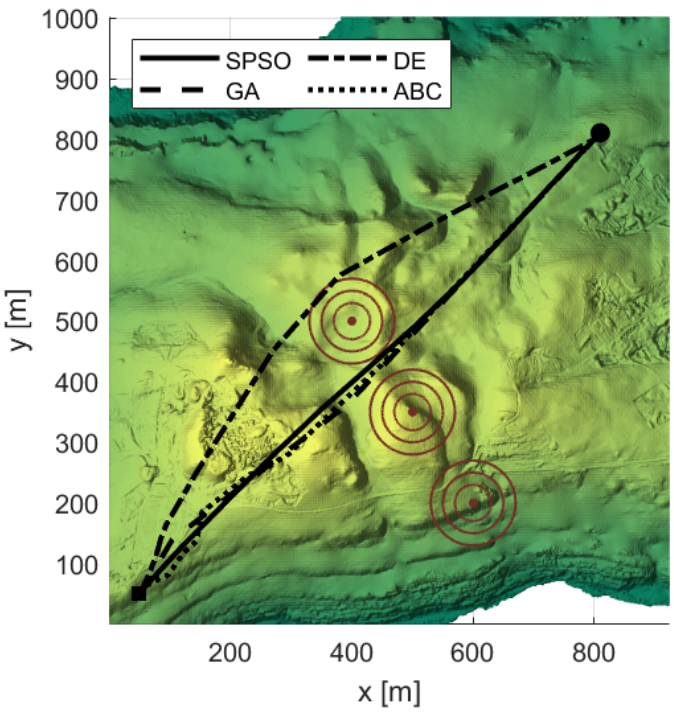


(d) Scenario 4

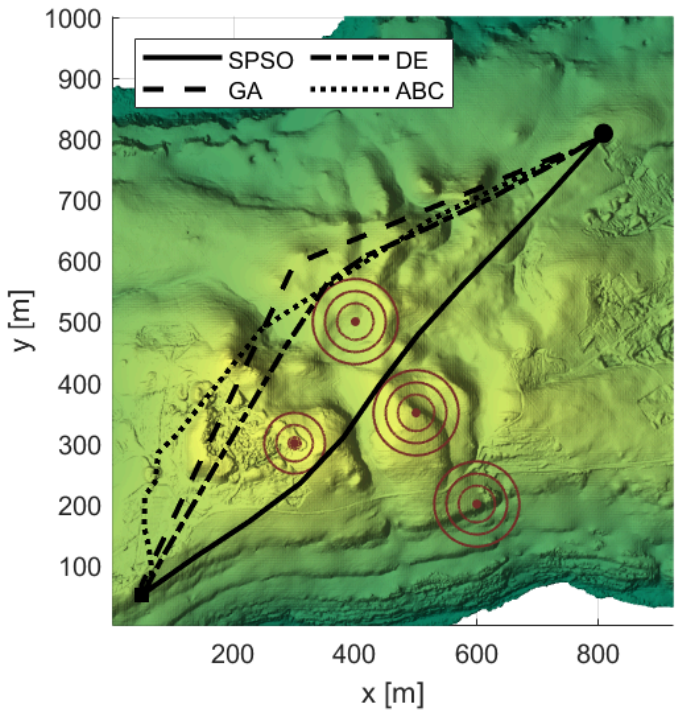
Figure 8: Top view of the paths generated by SPSO and other metaheuristic algorithms for scenarios 1 to 4



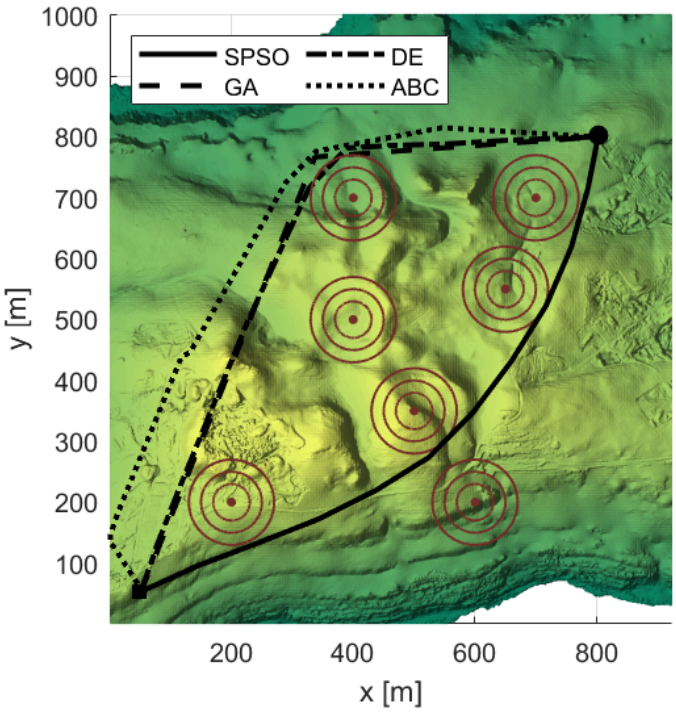
(a) Scenario 5



(b) Scenario 6

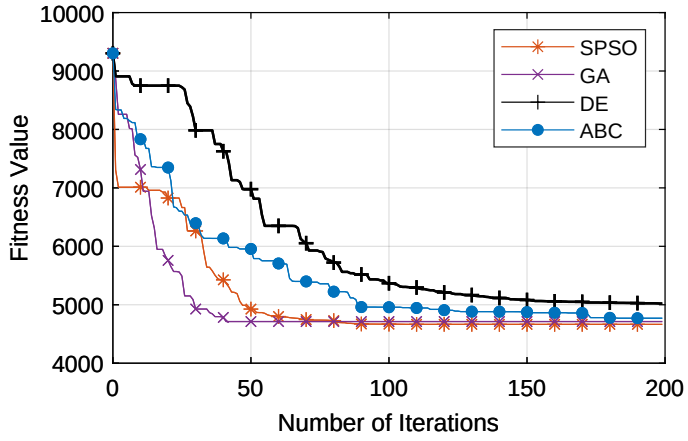


(c) Scenario 7

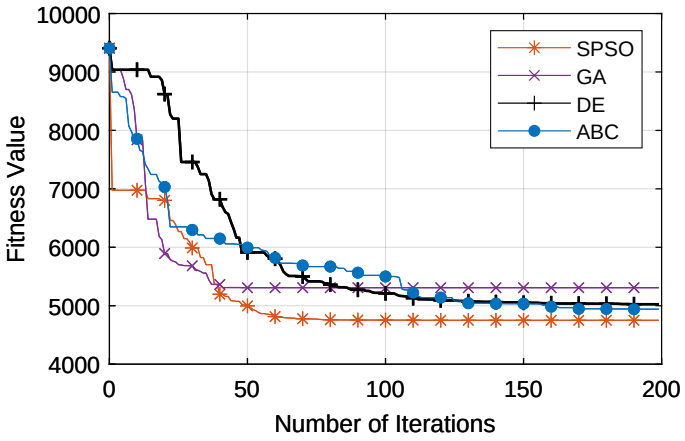


(d) Scenario 8

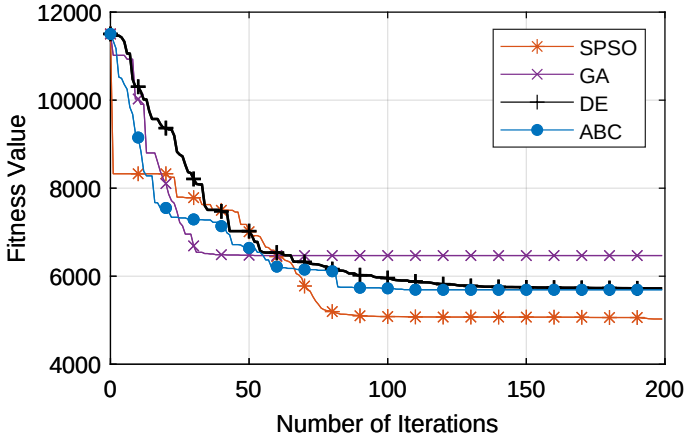
Figure 9: Top view of the paths generated by SPSO and other metaheuristic algorithms for scenarios 5 to 8



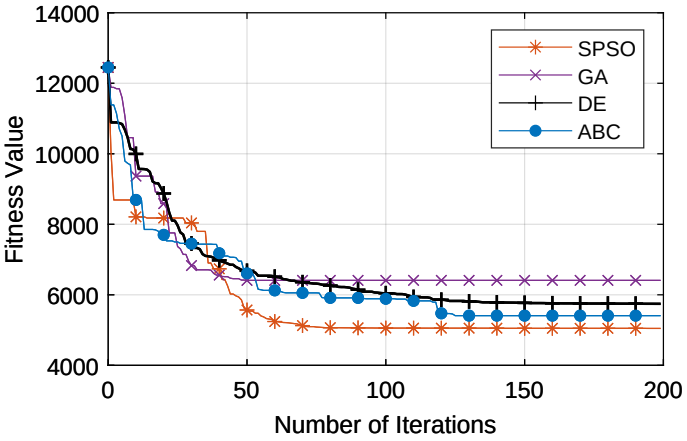
(a) Scenario 1



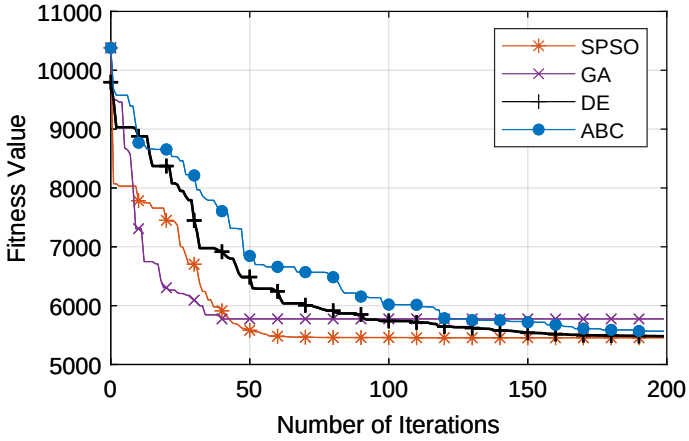
(b) Scenario 2



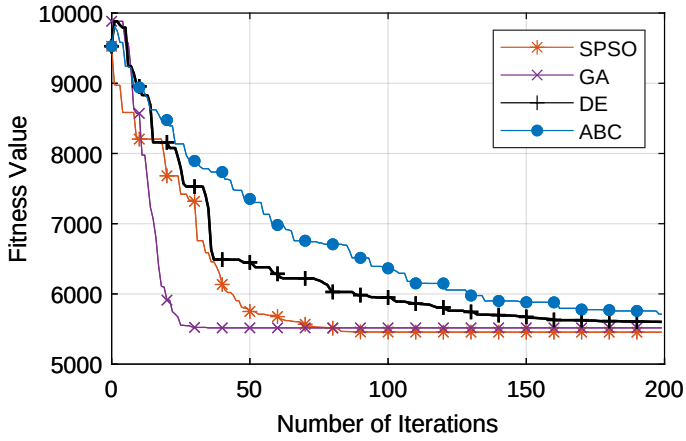
(c) Scenario 3



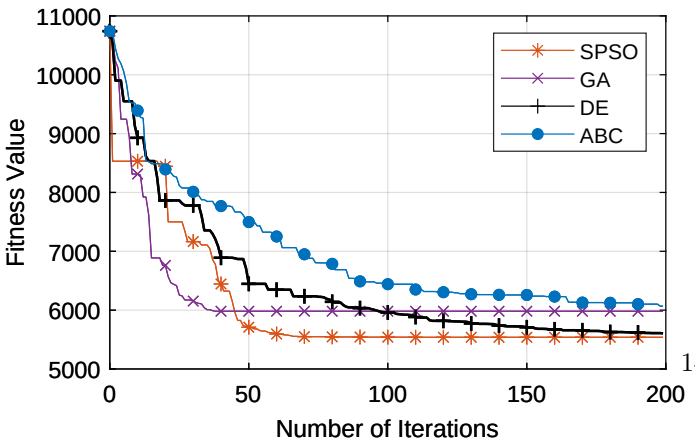
(d) Scenario 4



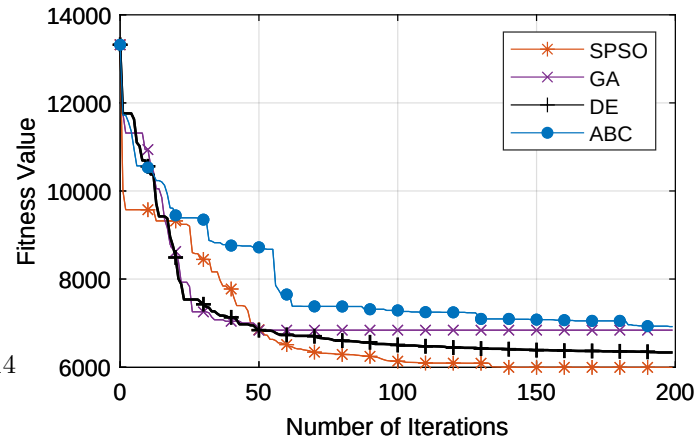
(e) Scenario 5



(f) Scenario 6

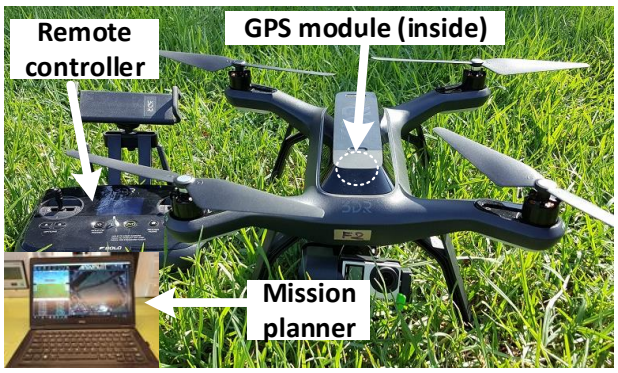


(g) Scenario 7

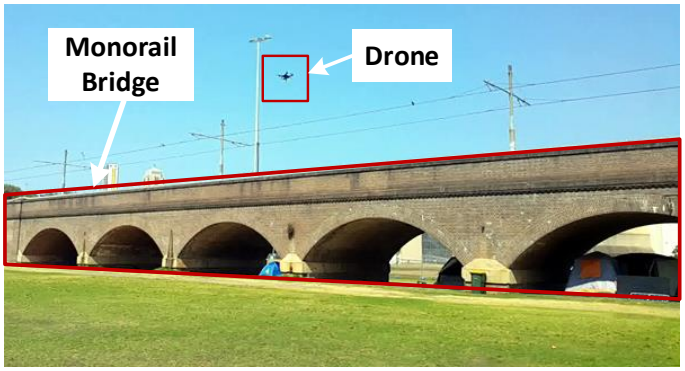


(h) Scenario 8

Figure 10: Best fitness values over iterations of SPSO and other metaheuristic algorithms

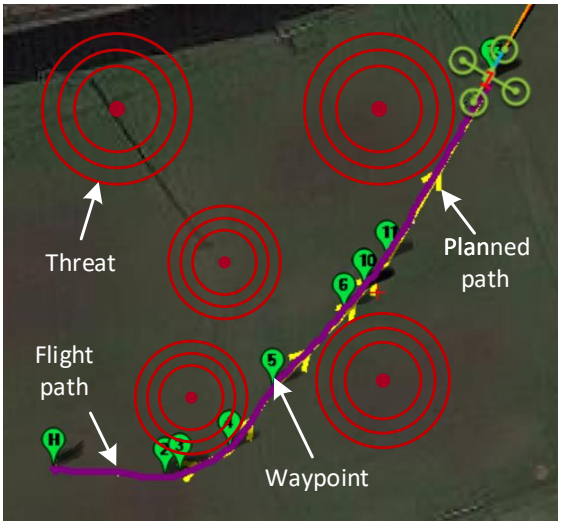


(a) 3DR Solo drone and Mission Planner software

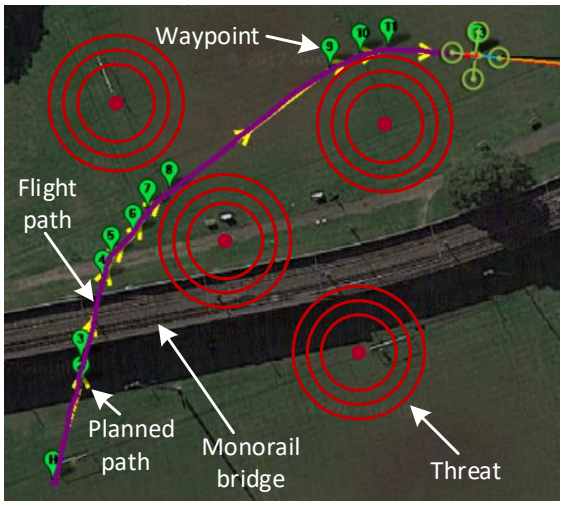


(b) Experimental field with a monorail bridge

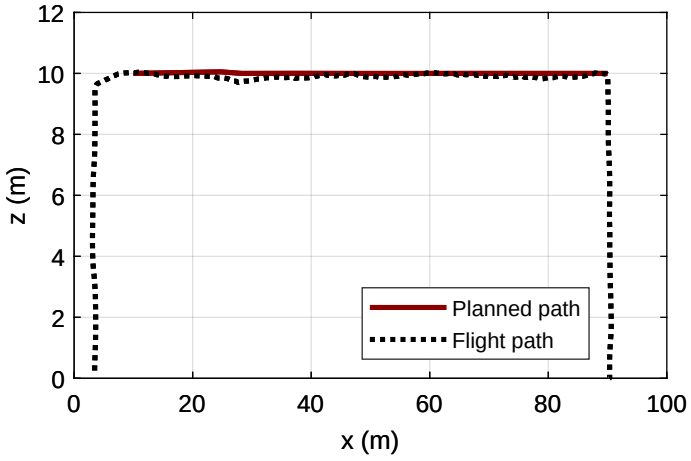
Figure 11: The drone and field used in experiments



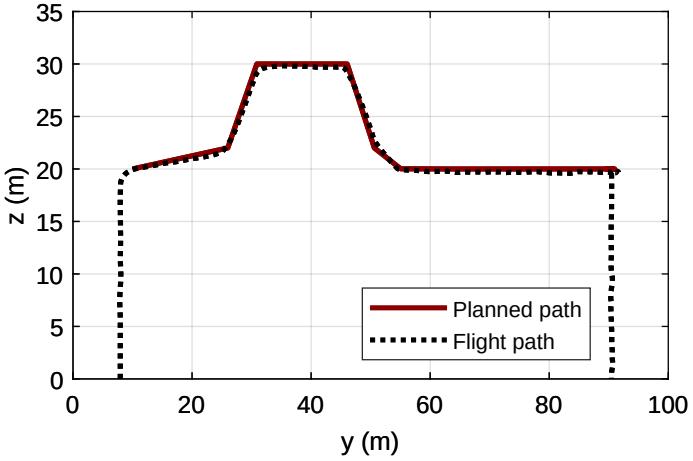
(a) The planned (yellow) and actual flight (magenta) paths in experimental scenario 1



(b) The planned (yellow) and actual flight (magenta) paths in experimental scenario 2



(c) Altitude of the planned and actual flight paths in experimental scenario 1



(d) Altitude of the planned and actual flight paths in experimental scenario 2

Figure 12: Experimental flight results

only the validity of the path planning algorithm but also the accuracy of the positioning system implemented in the drone.

5.5. Discussion

Through comparisons and experiments, it is clear that SPSO is capable of generating feasible, safe and optimal paths for UAV operation. The proposed algorithm performs especially well in complicated scenarios where many obstacles and threats appear reflected via its small fitness values and $D+$ t-test evaluations. The main drive for that effective performance rests with the idea of switching the search space, from the Cartesian to configuration space, where quality solutions can be obtained. Besides, constraints on UAV dynamics such as turning and climbing angles can be directly integrated into SPSO's variables to narrow down the search space. Nevertheless, hard constraints are being used in this study which may not be optimal for operations in which UAV states such as speed and altitude rapidly change over time.

In our design, the cost function is scalable in the sense that additional requirements like fuel consumption can be added as a term F_k with weight b_k to the overall cost function (9). However, choosing the right values of b_k to reflect the relationship among requirements may become complicated as the number of requirements increases. In those situations, multi-objective optimization can be considered to fulfill the task.

On another note, SPSO introduces relatively fast convergence as can be seen in Fig.6 and Fig.10 due to coherent interactions among particles. However, like many other PSO variants, it faces the problem of premature convergence where its particles converge to a local optimum in certain scenarios. This is the case with scenario 7 where DE performs better than SPSO due to its exploitation capability obtained via mutation and recombination. It suggests that a relevant randomization mechanism such as mutation, random walk or Lévy flight [48] may be useful to deal with the premature convergence problem.

6. Conclusion

We have presented a new algorithm, SPSO, for the problem of UAV path planning with the focus on the safety and feasibility of the paths generated. The cost function is designed so that the constraints associated with optimality, safety and feasibility are simultaneously incorporated. SPSO is developed based on the correspondence between intrinsic motion components of the UAV and the search space. Comparisons on eight benchmarking scenarios generated from DEM maps show that SPSO achieves the best quality paths in most scenarios. PSO and θ-PSO have stable convergence whereas QPSO only performs well for simple scenarios. Comparisons with other metaheuristic algorithms including GA, ABC, and DE also confirm the superior performance of SPSO. Experiments with real UAVs show the validity of the generated paths for practical operations. Besides, the correspondence between the particles of SPSO and UAV motion allows the kinematic constraints of UAV to be incorporated when necessary to further improve the path planning performance.

Our future work will focus on incorporating the exact constraints to be used for SPSO in the configuration space based on UAVs' dynamic model. We will also explore the applicability of SPSO to other optimization problems by evaluating its performance on different benchmarking functions.

References

[1] M. D. Phung, C. H. Quach, T. H. Dinh, Q. Ha, Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection, Automation in Construction 81 (2017) 25 – 33. doi:10.1016/j.autcon.2017.04.013.

[2] V. T. Hoang, M. D. Phung, T. H. Dinh, Q. P. Ha, System architecture for real-time surface inspection using multiple UAVs, IEEE Systems Journal 14 (2) (2020) 2925–2936.

[3] M. D. Phung, Q. P. Ha, Motion-encoded particle swarm optimization for moving target search using UAVs, Applied Soft Computing (2020) 106705doi:10.1016/j.asoc.2020.106705.

[4] L. Lin, M. A. Goodrich, UAV intelligent path planning for wilderness search and rescue, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 709–714.

[5] C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang, D. Wu, Offline and online search: UAV multiobjective path planning under dynamic urban environment, IEEE Internet of Things Journal 5 (2) (2018) 546–558.

[6] R. W. Beard, T. W. McLain, M. A. Goodrich, E. P. Anderson, Coordinated target assignment and intercept for unmanned air vehicles, IEEE Transactions on Robotics and Automation 18 (6) (2002) 911–922.

[7] T. W. McLain, R. W. Beard, Coordination variables, coordination functions, and cooperative timing missions, Journal of Guidance, Control, and Dynamics 28 (1) (2005) 150–161. doi:10.2514/1.5791.

[8] D. Eppstein, Finding the k shortest paths, SIAM Journal on Computing 28 (2) (1998) 652–673. doi:10.1137/S0097539795290477.

[9] P. O. Pettersson, P. Doherty, Probabilistic roadmap based path planning for an autonomous unmanned helicopter, Journal of Intelligent & Fuzzy Systems 17 (4) (2006) 395–405.

[10] Y. Lin, S. Saripalli, Sampling-based path planning for UAV collision avoidance, IEEE Transactions on Intelligent Transportation Systems 18 (11) (2017) 3179–3192.

[11] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.

[12] B. Penin, P. R. Giordano, F. Chaumette, Minimum-time trajectory planning under intermittent measurements, IEEE Robotics and Automation Letters 4 (1) (2019) 153–160.

[13] R. J. Szczerba, P. Galkowski, I. S. Glicktein, N. Ternullo, Robust algorithm for real-time route planning, IEEE Transactions on Aerospace and Electronic Systems 36 (3) (2000) 869–878.

[14] J. Li, G. Deng, C. Luo, Q. Lin, Q. Yan, Z. Ming, A hybrid path planning method in unmanned air/ground vehicle (UAV/UGV) cooperative systems, IEEE Transactions on Vehicular Technology 65 (12) (2016) 9585–9596.

[15] J. Kwak, Y. Sung, Autonomous UAV flight control for GPS-based navigation, IEEE Access 6 (2018) 37947–37955.

[16] X. Sun, Y. Liu, W. Yao, N. Qi, Triple-stage path prediction algorithm for real-time mission planning of multi-UAV, Electronics Letters 51 (19) (2015) 1490–1492.

[17] J. Barraquand, B. Langlois, J. . Latombe, Numerical potential field techniques for robot path planning, IEEE Transactions on Systems, Man, and Cybernetics 22 (2) (1992) 224–241.

[18] J. Tang, J. Sun, C. Lu, S. Lao, Optimized artificial potential field algorithm to multi-unmanned aerial vehicle coordinated trajectory planning and collision avoidance in three-dimensional environment, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 233 (16) (2019) 6032–6043. doi:10.1177/0954410019844434.

[19] G.-c. Luo, J.-q. Yu, Y.-s. Mei, S.-y. Zhang, UAV path planning in mixed-obstacle environment via artificial potential field method improved by additional control force, Asian Journal of Control 17 (5) (2015) 1600–1610. doi:10.1002/asjc.960.

[20] Y. bo Chen, G. chen Luo, Y. song Mei, J. qiao Yu, X. long Su, UAV path planning using artificial potential field method updated by optimal control theory, International Journal of Systems Science 47 (6) (2016) 1407–1420. doi:10.1080/00207721.2014.929191.

[21] H. Heidari, M. Saska, Collision-free trajectory planning of multirotor UAVs in a wind condition based on modified potential field, Mechanism and Machine Theory 156 (2021) 104140. doi:10.1016/j.mechmachtheory.2020.104140.

[22] B. Di, R. Zhou, H. Duan, Potential field based receding horizon motion planning for centrality-aware multiple UAV cooperative surveillance, Aerospace Science and Technology 46 (2015) 386 – 397. doi:10.1016/j.ast.2015.08.006.

[23] P.-C. Song, J.-S. Pan, S.-C. Chu, A parallel compact cuckoo search algorithm for three-dimensional path planning, Applied Soft Computing 94 (2020) 106443. doi:10.1016/j.asoc.2020.106443.

[24] V. Roberge, M. Tarbouchi, G. Labonte, Fast genetic algorithm path planner for fixed-wing military UAV using GPU, IEEE Transactions on Aerospace and Electronic Systems 54 (5) (2018) 2105–2117. doi:10.1109/TAES.2018.2807558.

[25] V. Roberge, M. Tarbouchi, G. Labonte, Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning, IEEE Transactions on Industrial Informatics 9 (1) (2013) 132–141. doi:10.1109/TII.2012.2198665.

[26] Y. Fu, M. Ding, C. Zhou, H. Hu, Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems 43 (6) (2013) 1451–1465. doi:10.1109/TSMC.2013.2248146.

[27] Z. Sun, J. Wu, J. Yang, Y. Huang, C. Li, D. Li, Path planning for GEO-UAV bistatic SAR using constrained adaptive multiobjective differential evolution, IEEE Transactions on Geoscience and Remote Sensing 54 (11) (2016) 6444–6457. doi:10.1109/TGRS.2016.2585184.

[28] C. Xu, H. Duan, F. Liu, Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning, Aerospace Science and Technology 14 (8) (2010) 535 – 541. doi:10.1016/j.ast.2010.04.008.

[29] X. Yu, W. Chen, T. Gu, H. Yuan, H. Zhang, J. Zhang, ACO-A*: Ant colony optimization plus a* for 3-D traveling in environments with dense obstacles, IEEE Transactions on Evolutionary Computation 23 (4) (2019) 617–631. doi:10.1109/TEVC.2018.2878221.

[30] Y. Fu, M. Ding, C. Zhou, Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 42 (2) (2012) 511–526. doi:10.1109/TSMCA.2011.2159586.

[31] J. Kennedy, R. Eberhart, Y. Shi (Eds.), Swarm Intelligence, Morgan Kaufmann, 2001. doi:10.1016/B978-1-55860-595-4.X5000-1.

[32] Zwe-Lee Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on Power Systems 18 (3) (2003) 1187–1195. doi:10.1109/TPWRS.2003.814889.

[33] R. C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: V. W. Porto, N. Saravanan, D. Waagen, A. E. Eiben (Eds.), Evolutionary Programming VII, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 611–616.

[34] S. Lalwani, H. Sharma, S. C. Satapathy, K. Deep, J. C. Bansal, A survey on parallel particle swarm optimization algorithms, Arabian Journal for Science and Engineering 44 (4) (2019) 2899–2923.

[35] P. Das, P. Jena, Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators, Applied Soft Computing 92 (2020) 106312. doi:10.1016/j.asoc.2020.106312.

[36] Y. Zhang, D. Gong, X. Sun, N. Geng, Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis, Soft Computing 18 (7) (2014) 1337–1352. doi:10.1007/s00500-013-1147-y.

[37] Y. Zhang, D. Gong, J. Zhang, Robot path planning in uncertain environment using multi-objective particle swarm optimization, Neurocomputing 103 (2013) 172–185. doi:10.1016/j.neucom.2012.09.019.

[38] N. Geng, X. Sun, D. Gong, Y. Zhang, Solving robot path planning in an environment with terrains based on interval multi-objective PSO, International Journal of Robotics and Automation 31 (2) (2016) 100–110. doi:10.2316/Journal.206.2016.2.206-4338.

[39] W.-M. Zhong, S.-J. Li, F. Qian, θ-PSO: a new strategy of particle swarm optimization, Journal of Zhejiang University-SCIENCE A 9 (6) (2008) 786–790. doi:10.1631/jzus.A071278.

[40] V. T. Hoang, M. D. Phung, T. H. Dinh, Q. P. Ha, Angle-encoded swarm optimization for UAV formation path planning, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 5239–5244. doi:10.1109/IROS.2018.8593930.

[41] J. Sun, W. Fang, X. Wu, V. Palade, W. Xu, Quantum-behaved particle swarm optimization: Analysis of individual particle behavior and parameter selection, Evolutionary Computation 20 (3) (2012) 349–393. doi:10.1162/EVCO_a_00049.

[42] M. Clerc, Discrete particle swarm optimization, illustrated by the traveling salesman problem, in: New optimization techniques in engineering, Springer, 2004, pp. 219–239.

[43] Geoscience Australia, Digital elevation model (DEM) of Australia derived from LiDAR 5 metre grid, Commonwealth of Australia and Geoscience Australia, Canberra (2015).

[44] H. Hsu, P. A. Lachenbruch, Paired t Test, American Cancer Society, 2005. doi:10.1002/0470011815.b2a15112.

[45] D. Karaboga, B. Basturk, On the performance of artificial bee colony (abc) algorithm, Applied Soft Computing 8 (1) (2008) 687 – 697. doi:10.1016/j.asoc.2007.05.007.

[46] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.

[47] Q. Li, S.-Y. Liu, X.-S. Yang, Influence of initialization on the performance of metaheuristic optimizers, Applied Soft Computing 91 (2020) 106193. doi:10.1016/j.asoc.2020.106193.

[48] H. Hakli, H. Uguz, A novel particle swarm optimization algorithm with levy flight, Applied Soft Computing 23 (2014) 333–345. doi:10.1016/j.asoc.2014.06.034.