Motion-encoded particle swarm optimization for moving target search using UAVs

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, Viet Nam*

ARTICLE INFO

ABSTRACT

This paper presents a novel algorithm named the motion-encoded particle swarm optimization (MPSO) for finding a moving target with unmanned aerial vehicles (UAVs). From the Bayesian theory, the search problem can be converted to the optimization of a cost function that represents the probability of detecting the target. Here, the proposed MPSO is developed to solve that problem by encoding the search trajectory as a series of UAV motion paths evolving over the generation of particles in a PSO algorithm. This motion-encoded approach allows for preserving important properties of the swarm including the cognitive and social coherence, and thus resulting in better solutions. Results from extensive simulations with existing methods show that the proposed MPSO improves the detection performance by 24% and time performance by 4.71 times compared to the original PSO, and moreover, also outperforms other state-of-the-art metaheuristic optimization algorithms including the artificial bee colony (ABC), ant colony optimization (ACO), genetic algorithm (GA), differential evolution (DE), and tree-seed algorithm (TSA) in most search scenarios. Experiments have been conducted with real UAVs in searching for a dynamic target in different scenarios to demonstrate MPSO merits in a practical application.

1. Introduction

Unmanned aerial vehicles (UAVs) have been receiving much research interest with numerous practical applications, especially in surveillance and rescue due to their capability of operating in harsh environments with sensor-rich work capacity suitable for different tasks. In searching for a lost target using UAVs, there often exists a critical period called "golden time" in which the probability the target being found should be highest [1]. As time progresses, that probability rapidly decreases due to the attenuation of initial information and the influence of external factors such as weather conditions, terrain features and target dynamics. The main objective in searching for a lost target using UAVs therefore includes finding a path that can maximize the probability of detecting the target within a specific flight time given initial information on target position and search conditions [2,3].

In the literature, the search problem is often formulated as probabilistic functions so that uncertainties in initial assumptions, search conditions and sensor models can be adequately incorporated. In [2,4], a Bayesian approach has been introduced to derive the objective functions for evaluating the detection probability of UAV flight paths. The initial search map has been modeled as a multivariate normal distribution with the mean and variance being computed based on initial information about the target position [5,6]. In [3,6], the target dynamic is represented by a stochastic Markov process which can then be deterministic or not depending on the searching scenarios. The sensor, on the other hand, is often modeled as either a binary variable with two states, "detected" or "not detected" [5], or as a continuous Gaussian variable [2].

Due to various probabilistic variables involved, the complexity of the searching problem varies from the level of nondeterministic polynomial-time hardness (NP-hard [7]) to nondeterministic exponential-time completeness (NEXP-complete [8]), in which the number of solutions available to search grows exponentially with respect to the search dimension and flight time. Consequently, solving this problem using classical methods such as differential calculus to find the exact solution becomes impractical, and hence, approximated methods are often used. A number of methods have been developed, such as greedy search with one-step look ahead [2] and k-step look ahead [3], ant colony optimization (ACO) [5], Bayesian optimization approach (BOA) [4], genetic algorithm (GA) [9,10], cross entropy optimization (CEO) [11], branch and bound approach [12], limited depth

search [13], and gradient descend methods [14,15]. Table 1 compares main properties of some algorithms where the "multi-agent" column implies the possibility of using multiple UAVs for searching and "ad hoc heuristic" for the case being specifically designated for the search problem. It is noted that most methods cope with moving targets and use the binary model for detection sensors. Some approaches [4,5,11,13] employ multiple UAVs to speed up the search process, whereas others use ad hoc heuristic to improve detection probability.

From the literature, it is recognizable that approaches to optimal search diverge in assumptions, constraints, target dynamics and searching mechanisms. Due to its complex nature, optimal search, especially in scenarios with fast-moving targets, remains a challenging problem. Besides, recent advancements in sensor, communication and UAV technologies enable the development of new search platforms. They pose the need for new methods that should not only robust in search capacity but also possess properties such as computational efficiency, adaptability and optimality.

For optimization, particle swarm optimization (PSO) is a potential technique with a number of key advantages that have been successfully applied in various applications [16–20]. It is less sensitive to initial conditions as well as the variation of objective functions and is able to adapt to many search scenarios via a small number of parameters including an initial weight factor and two acceleration coefficients [21]. It generally can find the global solution with a stable convergence rate and shorter computation time compared to other stochastic methods [22]. More importantly, PSO is simple in implementation with the capability of being parallelized to run with not only computer clusters or multiple processors but also graphical processing units (GPU) of a single graphical card. This allows to significantly reduce the execution time without requiring any change to the system hardware [23].

Motivated from the aforementioned analysis, we will employ the PSO methodology in this study to deal with the search problem in complex scenarios for fast moving targets, aiming to improve the search performance in both detection probability and execution time. To this end, we propose a new motion-encoded PSO algorithm, taking into account both cognitive and social coherence of the swarm. Our contributions include: (i) the formulation of an objective function for optimization, incorporating all assumptions and constraints, from the search problem and the probabilistic framework; (ii) the development of a new motion-encoded PSO (MPSO) from the idea of changing the search space for the swarm to avoid getting stuck at local maxima; (iii) the demonstration of MPSO implemented for UAVs in experimental search scenarios to validate its outperformance over other PSO algorithms obtained from extensive comparison analysis. The results show that MPSO, on one hand, presents superior performance on various search scenarios while on the other hand remains simple for practical implementation.

The rest of this paper is structured as follows. Section 2 outlines the steps to formulate the objective function. Section 3 presents the proposed MPSO and its implementation for solving a complex search problem. Section 4 provides simulation and experimental results. A conclusion is drawn in Section 5 to close our paper.

2. Problem formulation

The search problem is formulated by modeling the target, sensor and belief map with details as follows.

2.1. Target model

In the searching problem, the target is described by an unknown variable $x \in X$ representing its location. Before the search starts, a probability distribution function (PDF) is used to model the target location based on the available information, e.g., the last known location of the target before losing its signal. This PDF could be a normal distribution centered about the last known location, but also could be a uniform PDF if nothing is known about the target location. In the searching space, this PDF is represented by a grid map called the belief map, $b(x_0)$, in which the value in each cell corresponds to the probability of the target being in that cell. The map can be created by discretizing the searching space S into a grid of $S_r \times S_c$ cells and associating a probability to each cell. Assume the target presents in the searching space, we have $\sum_{x_0 \in S} b(x_0) = 1$.

During the searching process, the target may be not static but navigate in a certain pattern. This pattern can be modeled by a stochastic process which can be assumed as a Markov process. In the special case of a conditionally deterministic target, which is considered in this study, that pattern merely depends on the initial position x_0 of the target. In that case, the transition function, $p(x_t|x_{t-1})$, representing the probability which the target goes from cell x_{t-1} to x_t, is known for all cells $x_t \in S$. Consequently, the path of the target will be entirely known if its initial position is known. This assumption is made quite often for the survivor search at sea [24] and also for the search problems in general [5].

2.2. Sensor model

In order to look for and find a target, a sensor is installed on the UAV to carry out an observation z_t at each time step t. The observations are independent such that the occurrence of one observation provides no information about the occurrence of the other observation. A detection algorithm is implemented to return a result for each observation which is assumed to have only two possible outputs, the detection of the target, $z_t = D_t$, or no detection, $z_t = \bar{D}_t$, where D_t represents a "detection" event at time t. Due to imperfectness of the sensor and detection algorithm, an observation of the target detected, $z_t = D_t$, still does not ensure the presence of the target at x_t. This is reflected through the observation likelihood, $p(z_t|x_t)$, given knowledge of the sensor model. The likelihood of no detection, given a target location x_t, is then computed by:

$$p(\bar{D}_t|x_t) = 1 - p(D_t|x_t). \tag{1}$$

2.3. Belief map update

Once the initial distribution, $b(x_0)$, is initialized, the belief map of the target at time t, $b(x_t)$, can be established based on the Bayesian approach and the sequence of observations, $z_{1:t} = \{z_1, \ldots, z_t\}$, made by the sensor. This approach is conducted recursively via two phases, prediction and update. In the prediction, the belief map is propagated over time in accordance with the target motion model. Suppose at time t, the previous belief map, $b(x_{t-1})$, is available. Then, the predicted belief map is calculated as:

$$\hat{b}(x_t) = \sum_{x_{t-1} \in S} p(x_t|x_{t-1})b(x_{t-1}). \tag{2}$$

Notice from (2) that the belief map $b(x_{t-1})$ is in fact the conditional probability of the target being at x_{t-1} given observations up to $t - 1$, $b(x_{t-1}) = p(x_{t-1}|z_{1:t-1})$. When the observation z_t is available, the update is conducted simply by multiplying

Table 1
Comparison between search methods.

Method	Work	Target	Binary sensor	Multi-agent	Ad hoc heuristic
one-step look ahead	[2]	Static & Dynamic	✗	✗	✓
k-step look ahead	[3]	Dynamic	✓	✗	✓
BOA	[4]	Dynamic	✓	✓	✗
ACO	[5]	Dynamic	✓	✓	✓
GA	[10]	Static	✓	✗	✗
CEO	[11]	Dynamic	✓	✓	✗
Depth search	[13]	Static	✓	✓	✓
Gradient descent	[14]	Static	✗	✗	✗

the predicted belief map by the new conditional observation likelihood as follows:

$$b(x_t) = \eta_t p(z_t|x_t)\hat{b}(x_t), \tag{3}$$

where η_t is the normalization factor,

$$\eta_t = 1/\sum_{x_t \in S} p(z_t|x_t)\hat{b}(x_t). \tag{4}$$

η_t scales the probability that the target presents inside the searching area to one, i.e., $\sum_{x_t \in S} b(x_t) = 1$.

2.4. Searching objective function

According to the Bayesian theory, the probability that the target does not get detected at time t during an observation, $r_t = p(\bar{D}_t|z_{1:t-1})$, relies on two factors: (i) the latest belief map from the prediction phase (2), and (ii) the no detection likelihood (1). Across the whole searching area, that probability is given by:

$$r_t = \sum_{x_t \in S} p(\bar{D}_t|x_t)\hat{b}(x_t). \tag{5}$$

Notice that r_t is exactly the inverse of the normalization factor η_t in (4), $r_t = 1/\eta_t$, for a "no detection" event, $z_t = \bar{D}_t$, and thus is smaller than 1. By multiplying the not detected probability r_t over time, the joint probability of failing to detect the target from time 1 to t, $R_t = p(\bar{D}_{1:t})$, is then obtained:

$$R_t = \prod_{k=1}^{t} r_k = R_{t-1}r_t. \tag{6}$$

Hence, the probability that the target gets detected for the first time at time t is computed as:

$$p_t = \prod_{k=1}^{t-1} r_k(1-r_t) = R_{t-1}(1-r_t). \tag{7}$$

Summing p_t over t steps gives the probability of detecting the target in t steps:

$$P_t = \sum_{k=1}^{t} p_k = P_{t-1} + p_t. \tag{8}$$

P_t is thus often referred to as the "cumulative" probability to distinguish it with p_t. Notice that

$$P_t = 1 - R_t, \tag{9}$$

and as t grows, the probability of first detection p_t becomes smaller because the chance of detecting the target in previous steps increases. The cumulative probability P_t is thus bounded and increases toward one as t goes to infinity.

The objective function for the searching problem can now be formulated based on (8) given a finite search time. Let the search time period be $\{1, \ldots, N\}$, the goal of the searching strategy is to determine a search path $O = (o_1, \ldots, o_N)$ that could maximize the cumulative probability P_t. As such, the objective function is eventually formulated as follows:

$$J = \sum_{t=1}^{N} p_t. \tag{10}$$

3. Motion-encoded particle swarm optimization

As the search problem defined in (10) is NP-hard [7,8], the time required to calculate all possible paths to find the optimal solution would greatly increase and become intractable. Therefore, a heuristic approach like PSO can be a good option for solving the optimal search problem as in this study.

3.1. Particle swarm optimization

PSO is a population-based stochastic technique, inspired by social behavior of bird flocking, designed for solving optimization problems [16,25]. In PSO, a swarm of particles is initially generated with random positions and velocities. Each particle then moves and evolves in a cognitive fashion with other particles to seek the global optimum. Those movements are driven by its best position, L_k, and the best position of the swarm, G_k. Let x_k and v_k be the position and velocity of a particle at generation k, respectively. The movement of that particle in the next generation is given by:

$$v_{k+1} \leftarrow wv_k + \varphi_1 r_1(L_k - x_k) + \varphi_2 r_2(G_k - x_k) \tag{11}$$

$$x_{k+1} \leftarrow x_k + v_{k+1}, \tag{12}$$

where w is the inertial weight, φ_1 is the cognitive coefficient, φ_2 is the social coefficient, and r_1, r_2 are random sequences sampled from a uniform probability distribution in the range [0, 1]. From (11) and (12), the movement of a particle is directed by three factors, namely, following its own way, moving toward its best position, or moving toward the swarm's best position. The ratio among those factors is determined by the values of w, φ_1, and φ_2.

3.2. MPSO for optimal search

There have been several modifications and improvements from the PSO algorithm, depending on the application. However, the implementation of PSO for online searching for dynamic targets in a complex environment remains a challenging task, particularly in a limited time window. For the search problem, it is desired to encode the position of particles in a way that the particles can gradually move toward the global optimum. A common approach is to define a position as a multi-dimensional vector representing a possible search path:

$$x_k \sim O_k = (o_{k,1}, \ldots, o_{k,N}), \tag{13}$$

where $o_{k,i}$ corresponds to a node of the search map [26,27]. The drawback of this approach is that it does not cover the adjacent dynamic behavior in path nodes and thus may result in invalid
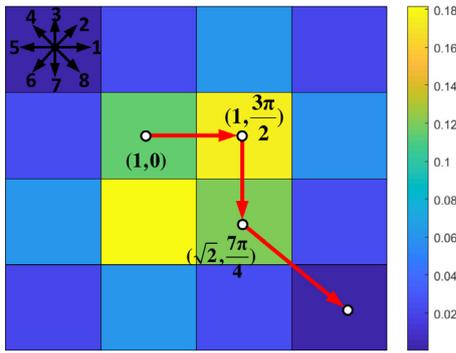
Fig. 1. Motion-encoded illustration for a path with three segments, $U_k = ((1, 0), (1, 3\pi/2), (\sqrt{2}, 7\pi/4))$.

paths during the searching process. Discrete PSO can be used to overcome this problem, but the momentum of particles is not preserved, causing local maxima [28]. Indirect approaches such as the angle-encoded PSO [29] and priority-based encoding PSO [30] can be a good option to deal with it and generate better results. Their mapping functions, however, require the phase angles to be within the range of $[-\pi/2, \pi/2]$ which limits the search capacity, especially in a large dimension.

Here, we propose the idea of using UAV motion to encode the position of particles. Instead of using nodes, we view each search path as a set of UAV motional segments, each corresponds to the movement of UAV from its current cell to another on the plane of flight. By respectively defining the magnitude and direction of the motion at time t as ρ_t and α_t, that motion can be completely described by a vector $u_t = (\rho_t, \alpha_t)$. A search path is then described by a vector of N motion segments, $U_k = (u_{k,1}, \ldots, u_{k,N})$. Using U_k as the position of each particle, equations for MPSO can be written as:

$$\Delta U_{k+1} \leftarrow wU_k + \varphi_1 r_1(L_k - U_k) + \varphi_2 r_2(G_k - U_k) \tag{14}$$

$$U_{k+1} \leftarrow U_k + \Delta U_{k+1}. \tag{15}$$

Fig. 1 illustrates a path with three segments, $U_k = ((1, 0), (1, 3\pi/2), (\sqrt{2}, 7\pi/4))$, where the belief map is color-coded with probability values indicated on the right.

During the search, it is also required to map U_k to a direct path O_k so that the cost associated with U_k can be evaluated. As shown in Fig. 1, the mapping process can be carried out by first constraining the UAV motion to one of its eight neighbors in each time step. Then, the motion magnitude ρ_t can be normalized and the motion angle α_t can be quantized as:

$$\rho_t^* = 1 \tag{16}$$

$$\alpha_t^* = 45° \lfloor \alpha_t/45° \rceil, \tag{17}$$

where $\lfloor \rceil$ represents the operator for rounding to the nearest integer. Node $o_{k,t+1}$ corresponding to the location of UAV in the Cartesian space is then given by:

$$o_{k,t+1} = o_{k,t} + u_{k,t}^*, \tag{18}$$

where

$$u_{k,t}^* = (\lfloor cos\alpha_t^* \rceil, \lfloor sin\alpha_t^* \rceil). \tag{19}$$

From the decoded path O_k, the cost value can be evaluated by the objective function (10) and then the local and global best can be computed as follows:

$$L_k = \begin{cases} U_k & \text{if } J(O_k) > J(L_{k-1}^*) \\ L_{k-1} & \text{otherwise} \end{cases}, \tag{20}$$

$$G_k = \underset{L_k}{\text{argmax}} J(O_k), \tag{21}$$

where L_k^* is the decoded path of L_k. It can be seen from the mapping process that (17) discretizes the motion to one of eight possible directions, (19) converts the moving direction to an increment in Cartesian coordinates, and (18) incorporates the increment to form the next node of the path.

Similarly to the interchange between the time domain and frequency domain in signal analysis, the mapping process of MPSO allows particles to search in the motion space instead of the Cartesian space. This leads to the following advantages:

- The motion space maintains the location of nodes consecutively so that the resultant paths after each generation evolvement are always valid, which is not the case of the Cartesian space;
- In motion space, the momentum of particles and swarm behaviors including exploration and exploitation are preserved so that the search performance is maintained and the swarm is able to cope with different target dynamics;
- As the normalization of ρ_t and quantization of α_t in (16) and (17) are only carried out for the purpose of cost evaluation, their continuous values are still being used for velocity and position updates as in (14)–(15). This property is important to avoid the discretizing effect of PSO so that the search resolution is not affected.

Finally, it is also noted that MPSO preserves the search mechanism of PSO via its update Eqs. (14)–(15) so that the advantages of PSO such as stable convergence, independence of initial conditions and implementation feasibility can be maintained.

3.3. Implementation

Fig. 2 shows the flowchart of MPSO to illustrate the implementation presented in Algorithm 1. Its structure is based on the core PSO but extended with the incorporation of the motion encoding and decoding steps. The belief map update as in (2) and (3) needs to be conducted during calculating the fitness when the target is non-static. Notably, the parallelism technique proposed in [23] can be applied to speed up the computation process of MPSO.

4. Results

To evaluate the performance of MPSO, we have conducted extensive simulation, comparison and experiments with detail described below.

4.1. Scenarios setup

For the sake of coverage, six different search scenarios are used to analyze the performance of MPSO for optimal search (some of them are adopted from [5]). The scenarios are defined to have the same map size ($S_r = S_c = 40$), but differ in the initial locations of UAV, target motion model $P(x_t|x_{t-1})$ and initial belief map $b(x_0)$. As shown in Fig. 3, the probability map is color-coded with the target dynamics presented by a white arrow and the initial location of UAV described by a white circle. The scenarios represent different searching situations as follows:

Scenario 1 has two high probability regions located next to each other. They are slightly different in location and value, which may cause difficulty in finding a better region to search for the target.

Scenario 2 includes two separated high probability regions located opposite to each other over the UAV location. The algorithm

Table 2
Comparison between PSO algorithms on fitness representing the accumulated detection probability.

Scenario	MPSO	PSO	QPSO	APSO
1	**0.1876 ± 0.0011**	0.1476 ±0.0043	0.1198 ± 0.0037	0.1869 ± 0.0025
2	**0.247 ± 0.0055**	0.2019 ± 0.0163	0.2014 ± 0.0046	0.2393 ± 0.0113
3	0.6554 ± 0.014	0.5403 ± 0.0218	0.5468 ± 0.014	**0.6649 ± 0.0287**
4	**0.5018 ± 0.0095**	0.4082 ± 0.0092	0.4259 ± 0.0164	0.4969 ± 0.0109
5	**0.2213 ± 0.0025**	0.1785 ± 0.0067	0.1819 ± 0.0008	0.2199 ± 0.004
6	**0.1881 ± 0.0112**	0.097 ± 0.0239	0.0943 ± 0.0168	0.1735 ± 0.0187

Fig. 2. Flowchart of MPSO algorithm.

```
/* Initialization:                          */
1  Get target dynamics and initial data;
2  Create belief map;
3  Set swarm parameters w, φ₁, φ₂, swarm_size;
4  foreach particle in swarm do
5  │    Create random motion-encoded paths Uₖ;
6  │    Assign Uₖ to particle position;
7  │    Compute fitness value of each particle;
8  │    Set local_best value of each particle to itself;
9  │    Set velocity of each particle to zero;
10 end
11 Set global_best to the best fit particle;
   /* Evolutions:                            */
12 for k ← 1 to max_generation do
13 │    foreach particle in swarm do
14 │    │    Compute motion velocity ΔUₖ₊₁;     /* Eq. (14) */
15 │    │    Compute new position Uₖ₊₁;         /* Eq. (15) */
16 │    │    Decode Uₖ₊₁ to Oₖ₊₁;              /* Eqs. (19)-(18) */
17 │    │    Update fitness of Oₖ₊₁;            /* Eq. (10) */
18 │    │    Update local_best Lₖ₊₁;            /* Eq. (20) */
19 │    end
20 │    Update global_best Gₖ₊₁;              /* Eq. (21) */
21 end
```

Algorithm 1: Pseudo code of MPSO.

Due to the stochastic nature of PSO, the algorithm is executed 10 times to find the average and standard deviation values for each scenario.

4.2. Search path

Fig. 4 shows the search paths of MPSO for each scenario together with the cumulative probability values. In all scenarios, MPSO is able to find the highest probability regions and generates relevant paths for the UAV to fly. For scenarios with only one high probability region such as Scenario 3 and 4, the cumulative probabilities are high because the chance of finding the target is not spread to other regions. It is also noted from Fig. 4 that the probability map only reflects the target belief at the last step whereas the search path represents the tracking of high probability regions over time. By comparing them with those in Fig. 3, we can see that the search paths adapt to the target dynamics to maximize the detection probability.

4.3. Comparison with other PSO algorithms

We have judged the merit of MPSO over other PSO algorithms including a classical PSO, denoted here as PSO for the comparison purpose, quantum-behaved PSO (QPSO) and angle-encoded PSO (APSO).

PSO is introduced in [16] in which the particles encode a search path as a set of nodes. They then evolve according to (11) and (12) to find the optimal solution.
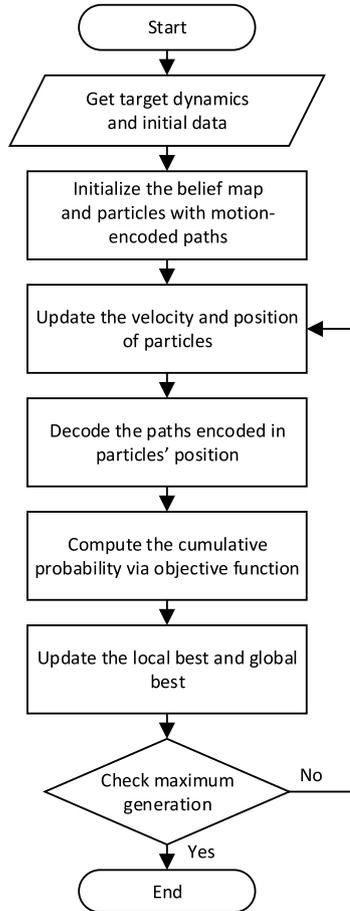
has to quickly identify the higher probability region to search and track as the target is moving south-west.

Scenario 3 has one small dense region moving rapidly toward the south-east. It thus tests the algorithm in its exploration and adaptation capability.

Scenario 4 is similar to Scenario 3 except that the target is moving toward the UAV's start location. It further evaluates the adaptability of the searching algorithm.

Scenario 5 consists of two probability regions located oppositely via the start location in which the right region is slightly higher in probability. As the target is moving north, the algorithm needs to identify the correct target region.

Scenario 6 is similar to Scenario 5, but the start location is below the potential regions and the target is moving North-East. It thus evaluates the capability of searching in a diagonal direction.

In our evaluations, MPSO is implemented with the parameters $w = 1$ at the damping rate of 0.98, $\varphi_1 = 2.5$ and $\varphi_2 = 2.5$. The swarm size is chosen to be 1000 particles. The number of iterations is 100 and the size of the search path is 20 nodes.
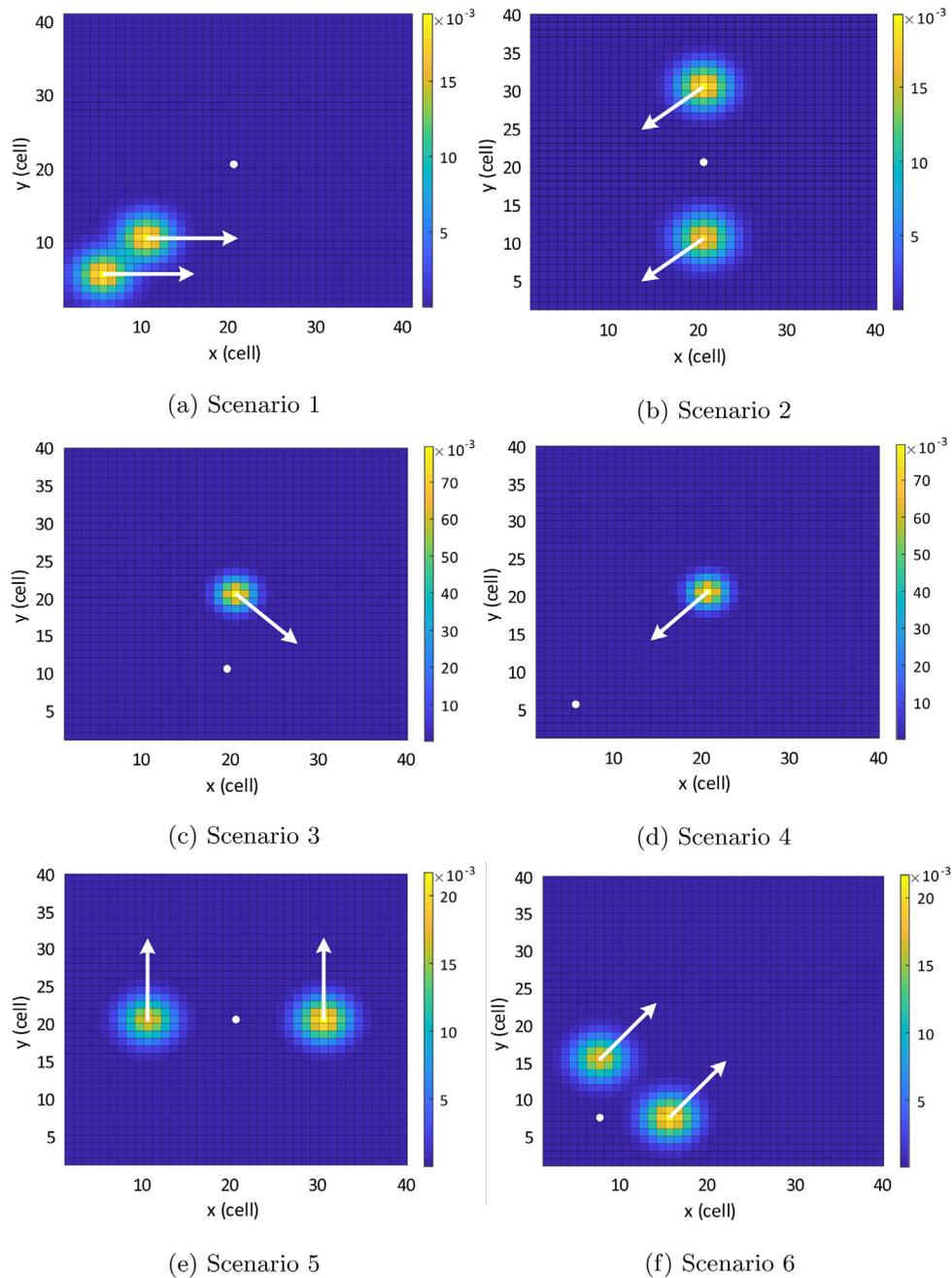
Fig. 3. Scenarios used for evaluating the searching algorithms.

APSO operates in a similar way as PSO. It, however, encodes the position of particles as a set of phase angles so that each angle represents the direction in which the path would emerge [29].

QPSO, on the other hand, assumes particles to have quantum behavior in a bound state. The particles are attracted by a quantum potential well centered on its local attractor and thus have a new stochastic update equation for their positions [31]. In QPSO, the position of particles also encodes a search path that includes a set of nodes.

Table 2 shows the average and standard deviation values of the fitness representing the accumulated detection probability obtained by all algorithms after 10 runs. It can be seen that MPSO introduces the best performance in 5 scenarios. APSO is slightly better than MPSO in Scenario 3, but its convergence is not stable

reflected via a larger standard deviation value. These results can be further verified via the convergence curves shown in Fig. 5. They show that PSO and QPSO present poor performance as the use of nodes to encode search paths does not maintain particle momentum resulting in local maxima.

APSO, on the other hand, introduces a comparable performance with MPSO. Unlike PSO and QPSO, the use of angles in APSO allows particles to search in orientation space and thus maintains the swarm properties. Interestingly, APSO can be considered as a special case of MPSO when the motion magnitude is constrained to 1. While this constraint limits the flexibility of the swarm, it may improve the exploration capacity in certain scenarios to yield a good result such as in Scenario 3.
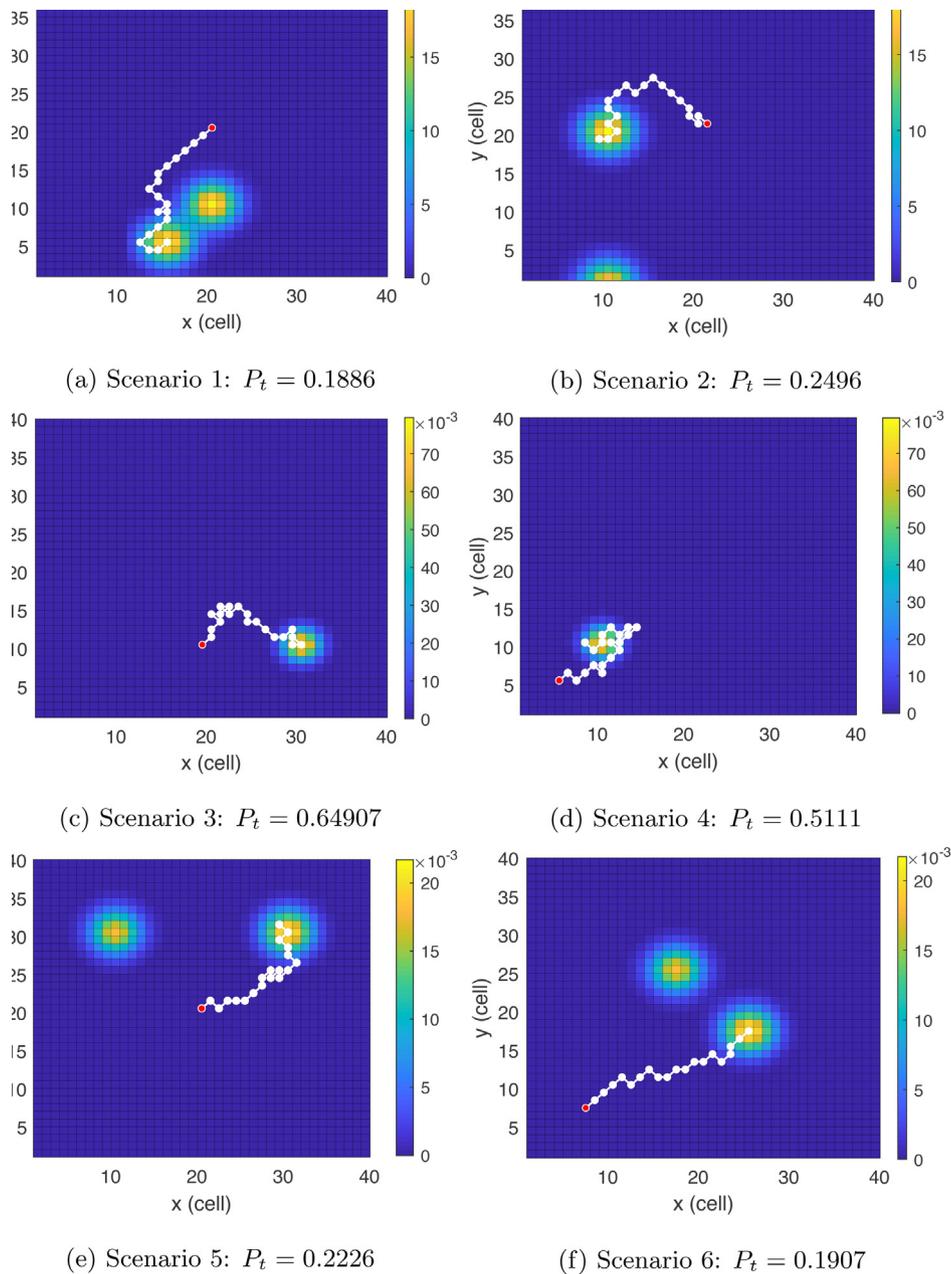
(a) Scenario 1: $P_t = 0.1886$

(b) Scenario 2: $P_t = 0.2496$

(c) Scenario 3: $P_t = 0.64907$

(d) Scenario 4: $P_t = 0.5111$

(e) Scenario 5: $P_t = 0.2226$

(f) Scenario 6: $P_t = 0.1907$

Fig. 4. Search paths for each scenario generated by MPSO.

4.4. Comparison with metaheuristic optimization algorithms

To further evaluate the performance of MPSO, we have compared it with state-of-the-art metaheuristic optimization algorithms including the artificial bee colony (ABC), ant colony optimization (ACO), genetic algorithm (GA), differential evolution (DE), and tree-seed algorithm (TSA).

ABC searches for optimal solutions based on the cooperative behavior of three types of bees: employed bees, onlooker bees and scout bees [32]. Our implementation represents each solution as a search path that consists of a set of motion segments similar to MPSO.

ACO solves optimization problems based on heuristic information and a pheromone model of artificial ants, each maintains a feasible solution [33]. Our implementation of ACO is based on [5] in which the "ACO-Node+H" approach is used together with the max–min ACO.

GA is a popular metaheuristic optimization that modifies a population of individual solutions similar to the process of natural selection [9]. Our implementation of GA is based on the "EA-dir" approach in [10] where a path is encoded as a string of directions subjected to two mutation techniques including "flip" and "pull".

DE is an optimization method that finds the optimal solution by improving its candidates via simple mathematical formulas from a population of individual solutions [34]. In implementing DE for optimal search, we represent each solution as a set of motions similar to the representation used in MPSO.

TSA solves the optimization problem by simultaneously exploring and exploiting the search space based on the spread of seeds from a tree population. The level and balance between the exploration and exploitation are controlled by predefined parameters including the search tendency (ST) and the number of seeds (NS). Those parameters are chosen as in the original study [35] in our implementation, i.e., $ST = 0.1$ and $NS \in [0.1, 0.25]$.

8   *M.D. Phung and Q.P. Ha / Applied Soft Computing Journal xxx (xxxx) xxx*

(a) Scenario 1

(b) Scenario 2

(c) Scenario 3
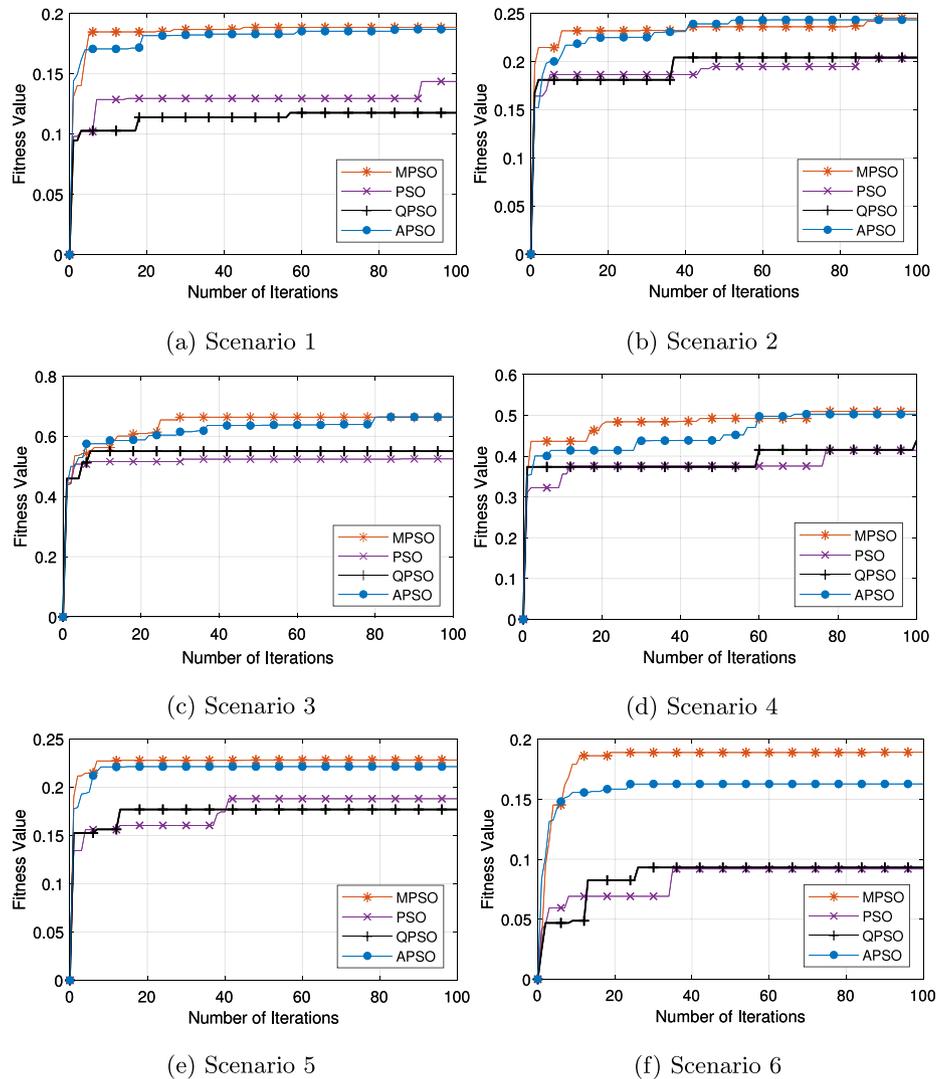
(d) Scenario 4

(e) Scenario 5

(f) Scenario 6

Fig. 5. Convergence curves of the four PSO algorithms on the six benchmark scenarios.

Table 3 presents the fitness values corresponding to the optimal solutions of MPSO and metaheuristic algorithms over six scenarios after 10 runs. The values include the average and standard deviation representing the cumulative detection probability. It can be seen that MPSO outperforms other metaheuristic algorithms in scenarios 1 to 5 with the highest fitness values and small standard deviation. TSA is the second best with satisfactory results in most scenarios, whereas the remaining algorithms are only good in one or two scenarios.

Fig. 6 further compares the convergence among the algorithms. While MPSO shows good exploitation capability represented via the high fitness value in most scenarios, its exploration reflected via the convergence speed is rather slow in some scenarios such as Scenario 3 where the high probability region is small and the target is moving away from the UAV. TSA, on the other hand, is good at exploration but rather limited in exploitation so that its final fitness values are slightly less than MPSO. ACO performs well in detecting static and slow-moving targets, but its adaptation to fast-moving targets is limited due to the nature of ACO incrementally exploring via nodes. DE and ABC have stable performance in most scenarios. GA, on the other hand, is often trapped at local minimums as the crossover and mutation operators cause many invalid paths during operation. Besides, the enhanced "flip" and "pull" operators which prioritize horizontal

and vertical search do not perform well in scenarios requiring diagonal search such as Scenario 6.

4.5. Execution time

Apart from the accuracy, we also evaluate the execution time of all algorithms to roughly estimate their complexity. We executed all algorithms under the same conditions of software and computer hardware. Table 4 shows the average execution time together with the standard deviation after 10 runs on an Intel Core i7-7600U 2.80 GHz processor. It can be seen that MPSO is the fastest in four scenarios, followed by ABC with two scenarios. DE also introduces relatively short execution time due to its simplicity in the search mechanism. TSA, on the other hand, is rather slow due to the extra computation required to evaluate the seeds of each tree. ACO is the slowest because of a large time spent on calculating heuristic information [5]. Notably, the execution time of APSO is close to MPSO which further explains it as a special case of MPSO. PSO and QPSO both require extra execution time due to the invalid paths generated during operation.

4.6. Validation on UAV platform

To demonstrate the practical use of MPSO, we have applied it to real searching scenarios with details as follows.

Table 3

Comparison between MPSO and other metaheuristic algorithms on fitness.

Scenario	MPSO	ABC	GA	ACO	DE	TSA
1	**0.1876 ± 0.0011**	0.1691 ± 0.0076	0.1283 ± 0.0001	0.1836 ± 0.0013	0.1818 ± 0.0015	0.1873 ± 0.0006
2	**0.247 ± 0.0055**	0.2099 ± 0.0041	0.2151 ± 0.0018	0.2145 ± 0.0049	0.22 ± 0.0045	0.2362 ± 0.0085
3	**0.6554 ± 0.014**	0.5872 ± 0.0152	0.5995 ± 0.003	0.6053 ± 0.02	0.5985 ± 0.0166	0.6236 ± 0.0135
4	**0.5018 ± 0.0095**	0.4225 ± 0.0017	0.3497 ± 0.0311	0.4866 ± 0.0139	0.4243 ± 0.0252	0.4626 ± 0.0239
5	**0.2213 ± 0.0025**	0.2093 ± 0.0071	0.1733 ± 0.0001	0.2208 ± 0.0024	0.2128 ± 0.006	0.2209 ± 0.0005
6	0.1881 ± 0.0112	0.181 ± 0.0019	0.1255 ± 0.0001	0.15 ± 0.0119	0.1829 ± 0.0139	**0.1889 ± 0.0018**

Table 4

Comparison between MPSO and other algorithms on execution time in seconds.

Scenario	MPSO	PSO	QPSO	APSO	ABC	GA	ACO	DE	TSA
1	43 ± 2	129 ± 6	140 ± 15	50 ± 8	**34 ± 1**	85 ± 2	144 ± 3	37 ± 3	84 ± 2
2	**26 ± 4**	150 ± 7	180 ± 22	34 ± 4	34 ± 5	95 ± 3	157 ± 2	32 ± 6	57 ± 6
3	**30 ± 8**	142 ± 4	149 ± 3	39 ± 4	31 ± 4	97 ± 1	150 ± 5	34 ± 2	50 ± 2
4	**20 ± 2**	149 ± 7	149 ± 1	32 ± 5	30 ± 3	92 ± 3	133 ± 3	26 ± 3	47 ± 1
5	**29 ± 7**	126 ± 4	129 ± 5	46 ± 5	34 ± 4	92 ± 3	150 ± 4	31 ± 3	60 ± 5
6	48 ± 7	140 ± 3	139 ± 2	61 ± 1	**39 ± 3**	99 ± 2	146 ± 13	**39 ± 3**	85 ± 2

(a) Scenario 1

(b) Scenario 2

(c) Scenario 3
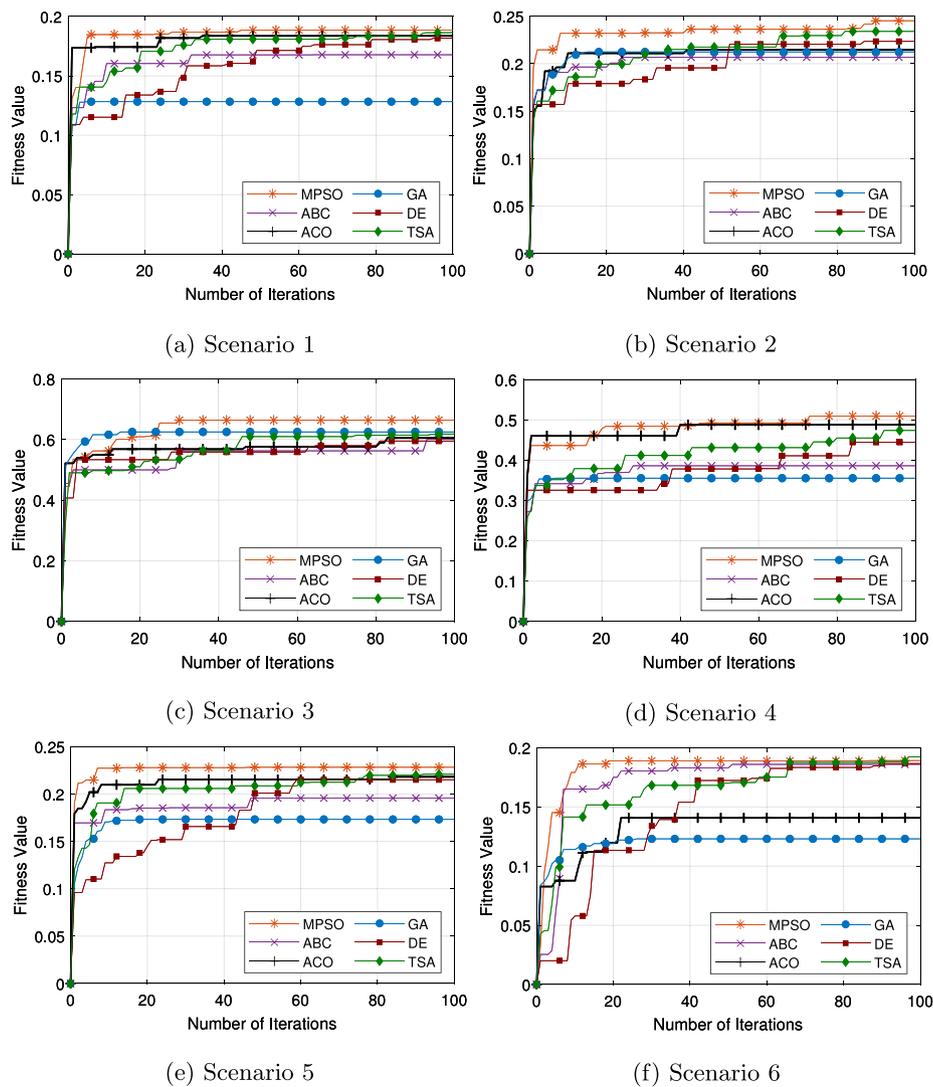
(d) Scenario 4

(e) Scenario 5

(f) Scenario 6

Fig. 6. Convergence curves of MPSO and other metaheuristic algorithms on the six benchmark scenarios.

4.6.1. Experimental setup

The experiment is carried out in the search area of 60 m × 60 m located in a park in Sydney. The UAV used is a 3DR Solo drone with a control architecture developed for infrastructure inspection [36] that can be controlled via a ground control station (GCS) software named Mission Planner. The detection sensor is a Hero 4 camera attached to the drone via a three-axis gimbal responsible for adjusting and stabilizing the camera. An unmanned ground vehicle (UGV) is used as the target. The UGV is equipped with
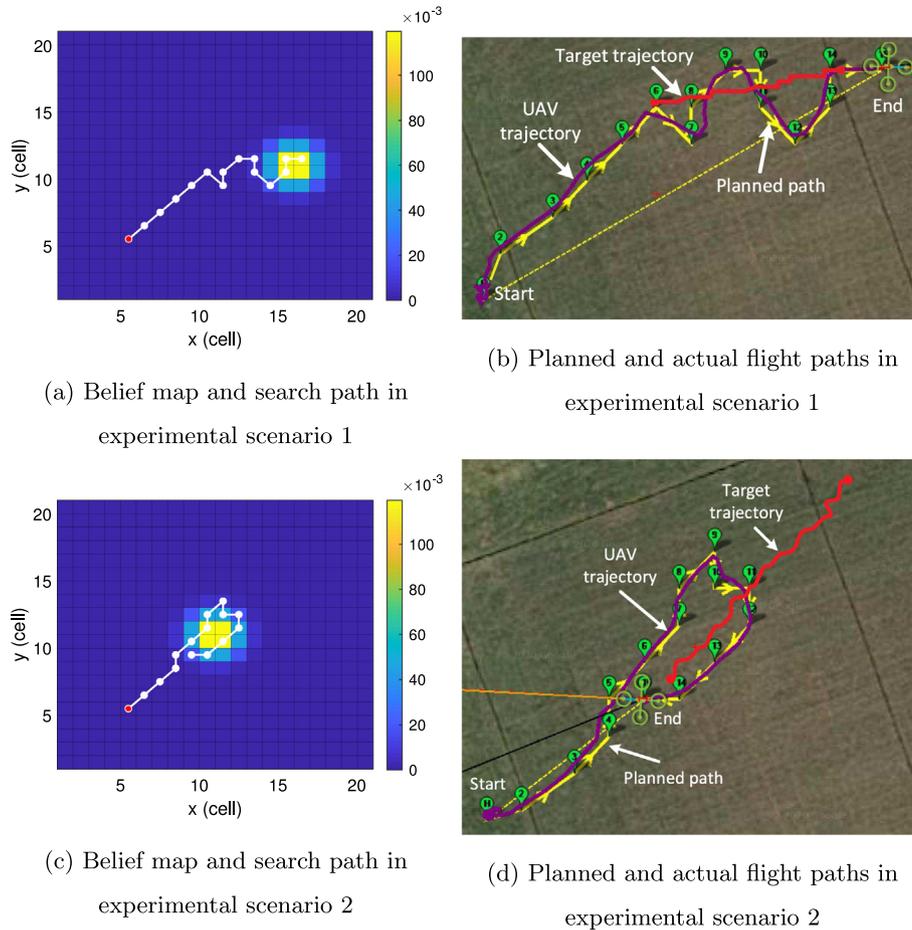
(a) Belief map and search path in experimental scenario 1

(b) Planned and actual flight paths in experimental scenario 1

(c) Belief map and search path in experimental scenario 2

(d) Planned and actual flight paths in experimental scenario 2

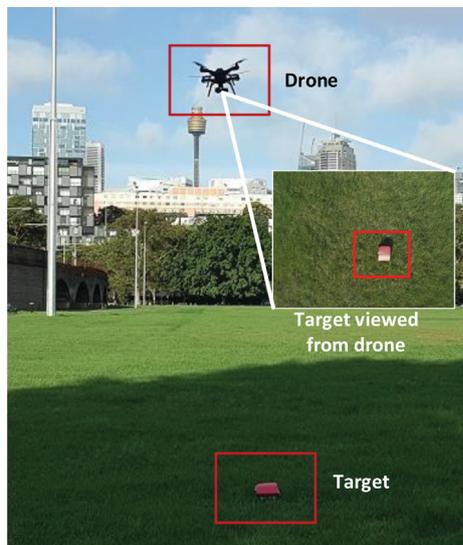**Fig. 7.** Experimental detection results.

Fig. 8. The target within the vision of the camera attached on the drone.

control and communication modules to allow it to track certain trajectories for the sake of experimental verification.

In experiments, initial locations of UAV and UGV are obtained via the GPS modules equipped on those vehicles and used as the input to generate a belief map. The map is fed to MPSO to generate a search path that includes a list of waypoints. Those waypoints are loaded into Mission Planner to fly the UAV. During the flight, for recording the testing results, positions of the vehicles are tracked via GPS and the video received from the camera is streamed to GCS.

4.6.2. Experimental results

Fig. 7(a) shows the belief map and path generated by MPSO for the scenario in which the UGV started from the center of the map at the latitude of -33.875992 and the longitude of 151.19145 and moved in East direction. Fig. 7(b) shows the planned and actual flight paths recorded via Mission Planner together with the actual path of UGV. It can be seen that the flight path tracks the planned path with some inevitably small tracking errors caused by GPS positioning. Those errors can be compensated for by extending the field of view of the detection camera via the flight attitude. The UAV thus can trace and approach the target at the location of $(-33.87598, 151.19153)$, as shown in Fig. 7(b). This can be verified in Fig. 8 that displays the target within the vision of the camera.

In another experiment where the UGV moves toward the starting location of the UAV, the planned path adapts to it by turning backward as shown in Fig. 7(c). Fig. 7(d) presents the actual trajectories of the UAV and UGV. It can be seen that the UAV tracks the planned path to approach the target at the location of $(-33.875938, 151.191515)$ and then can trace it eventually. Those results, together with various successful trials, confirm the validity and applicability of our proposed algorithm.

4.7. Discussion

Through extensive simulation, thorough comparison and experiments as described above, it can be seen that MPSO presents

better performance than other state-of-the-art heuristic algorithms in most search scenarios and is suitable for practical UAV search operations. The rationale for the success of MPSO lies in the motion-encoded mechanism that prevents the algorithm from generating invalid paths during the searching process so that it can avoid the need for re-initialization, and as such, to accelerate the convergence. The motion-encoded mechanism also allows MPSO to search in the motion space instead of the Cartesian space to improve search performance and better adapt to target dynamics. This advantage is clearly reflected in the good search result of MPSO for the challenging Scenario 4 where the target moves in the opposite direction to the search path that requires the UAV to turn around. Nevertheless, like PSO, MPSO may need to increase the swarm size and number of iterations if the search dimension increases [37]. In those scenarios, parallel implementation is required to effectively reduce the computation time, and hence, improve the scalability of the proposed algorithm for large-scale systems.

In practical search, the target dynamics may vary depending on the applications so that the deterministic assumption used in this study may go beyond its validity. In those scenarios, a prediction mechanism using optimal estimators such as the Kalman filter [38] can be employed to provide a prediction of the target trajectory. It is then used to calculate the cumulative probability used in the objective function of MPSO.

5. Conclusion

We have presented a new algorithm, the motion-encoded particle swarm optimization (MPSO), to solve the problem of optimal search for a moving target using UAVs. The algorithm encodes the search path as a series of motions that are directly applicable to the search problem which constrains the movement of a UAV to its neighbor cells. By changing the search domain from the Cartesian space to motion space, the algorithm is able to adapt to different target dynamics. It also preserves key properties of PSO to enhance the search performance and allows to conduct continuous search in discrete maps. Simulation and experimental results show that the algorithm is effective and practical enough to deploy for search operations. To be effective also for large-scale systems, the proposed algorithm would need parallel computation to further reduce its execution time. Our future work will focus on evaluating MPSO on benchmarking functions and exploring its capability to solve other complex optimization problems.

CRediT authorship contribution statement

Manh Duong Phung: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing - original draft, Visualization, Investigation. **Quang Phuc Ha:** Supervision, Writing - review & editing.

Declaration of competing 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.

References

[1] S.F. Ochoa, R. Santos, Human-centric wireless sensor networks to improve information availability during urban search and rescue activities, Inf. Fusion 22 (2015) 71–84, http://dx.doi.org/10.1016/j.inffus.2013.05.009.

[2] F. Bourgault, T. Furukawa, H.F. Durrant-Whyte, Optimal search for a lost target in a Bayesian world, in: S. Yuta, H. Asama, E. Prassler, T. Tsubouchi, S. Thrun (Eds.), Field and Service Robotics: Recent Advances in Reserch and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 209–222, http://dx.doi.org/10.1007/10991459_21.

[3] M. Raap, S. Meyer-Nieberg, S. Pickl, M. Zsifkovits, Aerial vehicle search-path optimization: A novel method for emergency operations, J. Optim. Theory Appl. 172 (3) (2017) 965–983, http://dx.doi.org/10.1007/s10957-016-1014-y.

[4] P. Lanillos, J. Yañez Zuluaga, J.J. Ruz, E. Besada-Portas, A Bayesian approach for constrained multi-agent minimum time search in uncertain dynamic domains, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13, ACM, New York, NY, USA, 2013, pp. 391–398, http://dx.doi.org/10.1145/2463372.2463417.

[5] S. Perez-Carabaza, E. Besada-Portas, J.A. Lopez-Orozco, J.M. de la Cruz, Ant colony optimization for multi-UAV minimum time search in uncertain domains, Appl. Soft Comput. 62 (2018) 789–806, http://dx.doi.org/10.1016/j.asoc.2017.09.009.

[6] T. Furukawa, F. Bourgault, B. Lavis, H.F. Durrant-Whyte, Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, 2006, pp. 2521–2526, http://dx.doi.org/10.1109/ROBOT.2006.1642081.

[7] K.E. Trummel, J.R. Weisinger, Technical note - The complexity of the optimal searcher path problem, Oper. Res. 34 (2) (1986) 324–327, http://dx.doi.org/10.1287/opre.34.2.324.

[8] D.S. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of Markov decision processes, Math. Oper. Res. 27 (4) (2002) 819–840, http://dx.doi.org/10.1287/moor.27.4.819.297.

[9] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, first ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[10] 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, http://dx.doi.org/10.1109/IROS.2009.5354455.

[11] P. Lanillos, E. Besada-Portas, G. Pajares, J.J. Ruz, Minimum time search for lost targets using cross entropy optimization, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 602–609, http://dx.doi.org/10.1109/IROS.2012.6385510.

[12] J.N. Eagle, J.R. Yee, An optimal branch-and-bound procedure for the constrained path, moving target search problem, Oper. Res. 38 (1) (1990) 110–114, http://dx.doi.org/10.1287/opre.38.1.110.

[13] A. Sarmiento, R. Murrieta-Cid, S. Hutchinson, An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments, Adv. Robot. 23 (12–13) (2009) 1533–1560, http://dx.doi.org/10.1163/016918609X12496339799170.

[14] S.K. Gan, S. Sukkarieh, Multi-UAV target search using explicit decentralized gradient-based negotiation, in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 751–756, http://dx.doi.org/10.1109/ICRA.2011.5979704.

[15] G. Mathews, H. Durrant-Whyte, M. Prokopenko, Asynchronous gradient-based optimisation for team decision making, in: 2007 46th IEEE Conference on Decision and Control, 2007, pp. 3145–3150, http://dx.doi.org/10.1109/CDC.2007.4434301.

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

[17] K.Y. Lee, J. Park, Application of particle swarm optimization to economic dispatch problem: Advantages and disadvantages, in: 2006 IEEE PES Power Systems Conference and Exposition, 2006, pp. 188–192.

[18] V. Mohammadi, S. Ghaemi, H. Kharrati, PSO tuned FLC for full autopilot control of quadrotor to tackle wind disturbance using bond graph approach, Appl. Soft Comput. 65 (2018) 184–195, http://dx.doi.org/10.1016/j.asoc.2018.01.015.

[19] T. Niknam, M.R. Narimani, M. Jabbari, Dynamic optimal power flow using hybrid particle swarm optimization and simulated annealing, Int. Trans. Electr. Energy Syst. 23 (7) (2013) 975–1001, http://dx.doi.org/10.1002/etep.1633.

[20] T. Niknam, M.R. Narimani, J. Aghaei, R. Azizipanah-Abarghooee, Improved particle swarm optimisation for multi-objective optimal power flow considering the cost, loss, emission and voltage stability index, IET Gener. Transm. Distrib. 6 (6) (2012) 515–527.

[21] 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, Vol. VII, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 611–616.

[22] Zwe-Lee Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Trans. Power Syst. 18 (3) (2003) 1187–1195.

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

[24] K. Iida, R. Hohzaki, K. Inada, Optimal survivor search for a target with conditionally deterministic motion under reward criterion, J. Oper. Res. Soc. Japan 41 (2) (1998) 246–260, http://dx.doi.org/10.15807/jorsj.41.246.

[25] M. Kohler, M.M. Vellasco, R. Tanscheit, PSO+: A new particle swarm optimization algorithm for constrained problems, Appl. Soft Comput. 85 (2019) 105865, http://dx.doi.org/10.1016/j.asoc.2019.105865.

[26] V. Roberge, M. Tarbouchi, G. Labonte, Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning, IEEE Trans. Ind. Inf. 9 (1) (2013) 132–141, http://dx.doi.org/10.1109/TII.2012.2198665.

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

[28] M. Clerc, Discrete particle swarm optimization, illustrated by the travelling salesman problem, in: New Optimization Techniques in Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 219–239, http://dx.doi.org/10.1007/978-3-540-39930-8_8.

[29] Y. Fu, M. Ding, C. Zhou, Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV, IEEE Trans. Syst. Man Cybern. A 42 (2) (2012) 511–526, http://dx.doi.org/10.1109/TSMCA.2011.2159586.

[30] A.W. Mohemmed, N.C. Sahoo, T.K. Geok, Solving shortest path problem using particle swarm optimization, Appl. Soft Comput. 8 (4) (2008) 1643–1653, http://dx.doi.org/10.1016/j.asoc.2008.01.002, Soft Computing for Dynamic Data Mining.

[31] J. Sun, W. Fang, X. Wu, V. Palade, W. Xu, Quantum-behaved particle swarm optimization: Analysis of individual particle behavior and parameter selection, Evol. Comput. 20 (3) (2012) 349–393, http://dx.doi.org/10.1162/EVCO_a_00049, PMID: 21905841.

[32] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697, http://dx.doi.org/10.1016/j.asoc.2007.05.007.

[33] M. Dorigo, T. Stützle, The ant colony optimization metaheuristic: Algorithms, applications, and advances, in: F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Springer US, Boston, MA, 2003, pp. 250–285, http://dx.doi.org/10.1007/0-306-48056-5_9.

[34] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.

[35] M.S. Kiran, TSA: Tree-seed algorithm for continuous optimization, Expert Syst. Appl. 42 (19) (2015) 6686–6698, http://dx.doi.org/10.1016/j.eswa.2015.04.055.

[36] V.T. Hoang, M.D. Phung, T.H. Dinh, Q.P. Ha, System architecture for real-time surface inspection using multiple UAVs, IEEE Syst. J. (2019) 1–12, http://dx.doi.org/10.1109/JSYST.2019.2922290.

[37] S. Piccand, M. O'Neill, J. Walker, On the scalability of particle swarm optimisation, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 2008, pp. 2505–2512.

[38] H. Musoff, P. Zarchan, Fundamentals of Kalman Filtering: A Practical Approach, American Institute of Aeronautics and Astronautics, 2009.