Phase Angle-Encoded and Quantum-Behaved Particle Swarm Optimization Applied to Three-Dimensional Route Planning for UAV

Yangguang Fu, *Student Member, IEEE*,
Mingyue Ding, *Senior Member, IEEE*, and Chengping Zhou

***Abstract*—A new variant of particle swarm optimization (PSO), named phase angle-encoded and quantum-behaved particle swarm optimization (θ-QPSO), is proposed. Six versions of θ-QPSO using different mappings are presented and compared through their application to solve continuous function optimization problems. Several representative benchmark functions are selected as testing functions. The real-valued genetic algorithm (GA), differential evolution (DE), standard particle swarm optimization (PSO), phase angle-encoded particle swarm optimization (θ-PSO), quantum-behaved particle swarm optimization (QPSO), and θ-QPSO are tested and compared with each other on the selected unimodal and multimodal functions. To corroborate the results obtained on the benchmark functions, a new route planner for unmanned aerial vehicle (UAV) is designed to generate a safe and flyable path in the presence of different threat environments based on the θ-QPSO algorithm. The PSO, θ-PSO, and QPSO are presented and compared with the θ-QPSO algorithm as well as GA and DE through the UAV path planning application. Each particle in swarm represents a potential path in search space. To prune the search space, constraints are incorporated into the pre-specified cost function, which is used to evaluate whether a particle is good or not. Experimental results demonstrated good performance of the θ-QPSO in planning a safe and flyable path for UAV when compared with the GA, DE, and three other PSO-based algorithms.**

***Index Terms*—Continuous function optimization, phase angle-encoded and quantum-behaved particle swarm optimization (θ-QPSO), route planning, unmanned aerial vehicle (UAV).**

I. Introduction

Route planning for unmanned aerial vehicle (UAV) is one of the most important parts of mission planning. Route planning is to generate a path between an initial prescribed location and a desired destination having an optimal or near-optimal performance under specific constraints [1]–[3]. Mathematically, the problem of 3-D route planning for UAV can be described as follows [4]:

Given the launching site A and target site $B, (A, B \in \mathbf{R}^3)$, K threat sets $\{T_1, T_2, \ldots, T_K\}$, and the parameters of UAV's maneuvering performance constraints (such as the restrictions of turning angle α, climbing/diving angle β, and flying height h, etc.), find a set of waypoints $\{W_0, W_1, \ldots, W_n, W_{n+1}\}$ with $W_0 = A$ and $W_{n+1} = B$ such that the resultant path is safe and flyable. In other words, for the reported path, no line segment intersects the interior of any $T_1, T_2, \ldots, T_K$ and all constraints are satisfied.

In recent years, route planning for UAV in both military and civilian applications has been widely researched. The military applications of UAV include reconnaissance, surveillance, battle damage assessment, target prosecution, and so on. Meanwhile, there are numerous civilian uses for UAV including fire fighting, disaster relief, search and rescue, as well as space-oriented applications. The existing approaches of route planning for UAV can be classified as follows:

1) Graph-based algorithms. The Voronoi diagram and its dual the Delaunay triangulation have been widely used in robot path planning [5] and UAV route planning. In [6] and [7], the threat regions are modeled as points, and these point threats are used to generate the Voronoi diagram. The Voronoi diagram partitions the battle area into several convex polygons, which contain one and only one threat. The graph containing the start and destination locations is searched using Eppstein's k-best paths algorithm [8] to find the optimal path for UAV. Another commonly used graph-based algorithm is the probabilistic roadmap. Pettersson and Doherty [9] designed a probabilistic roadmap-based planner for an autonomous unmanned helicopter. However, it is difficult to combine the motion constraints of UAV with graph-based algorithms, unless the maneuverability of UAV is good enough. Therefore, graph-based algorithms are usually used for 2-D path planning.
2) Heuristic search algorithms. A* is a best-first heuristic search algorithm that finds the least-cost path from a given initial node to one goal node and was first described by Hart *et al.* [10]. On the basis of Hart's work, Szczerba *et al.* proposed the Sparse A* Search (SAS) algorithm [11] to plan a real-time route for aircraft. SAS approach accurately and efficiently prunes the search space since various route constraints are taken into consideration during the planning process. A* and SAS planned a path based on known environment information, incapable of planning paths in unknown, partially known, and changing environments. To solve this problem, Stentz [12] first introduced the D* search algorithm. Unfortunately, as the problem space becomes larger and larger, the time spent to find the optimal or near-optimal path increases exponentially, whether A* or D* algorithm is adopted.
3) Evolutionary computation algorithms. Four well-known paradigms of evolutionary computation (EC) are genetic algorithms (GA), evolutionary programming (EP), evolutionary strategies (ES), and genetic programming (GP) [13]. It is generally accepted that EC is an efficient and effective optimization technique and has been used as a viable candidate to solve path planning problems. Yi *et al.* [14] proposed a 3-D route planning method based on GA. Zheng *et al.* [1]–[3] and Nikolos *et al.* [15], [16] both present a 2-D or 3-D route planner for UAV using evolutionary algorithms. The difference is the way of getting a smooth route. To get a smooth route, Zheng *et al.* incorporated the turning angle constraint into the search process while Nikolos *et al.* used B-Spline curves with the coordinates of its control points being the chromosome genes of evolutionary algorithms. Each chromosome (path) consists of the same fixed number of genes, i.e., the coordinates of B-Spline control points. An in-depth discussion of other existing algorithms such as dynamic programming, simulated annealing, and artificial potential field for the route planning-oriented application can be found in [3].

Particle swarm optimization (PSO) is a new evolutionary computation algorithm first proposed by Kennedy and Eberhart [17]. Its development is based on the study of the social behavior of bird flocking and fish schooling. Similar to other evolutionary algorithms such as GA [13] and differential evolution (DE) [18], [19], PSO is a population-based optimization algorithm. However, like ant colony optimization (ACO) [20] and some estimation of distribution algorithms

Manuscript received November 16, 2009; revised May 28, 2010, August 8, 2010, October 8, 2010, and January 6, 2011; accepted March 6, 2011. Date of publication July 7, 2011; date of current version February 17, 2012. This paper was recommended by Associate Editor K. Hirota.

Y. Fu and C. Zhou are with the State Key Laboratory for Multi-spectral Information Processing Technologies, Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yangguangfu@yahoo.com.cn; zhouchp@sina.com).

M. Ding is with the "Image Information Processing and Intelligence Control" Key Laboratory of Education Ministry of China, College of Life Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: myding@mail.hust.edu.cn).

Digital Object Identifier 10.1109/TSMCA.2011.2159586

(EDAs) [21], PSO has two salient characteristics strikingly different from other evolutionary algorithms. Firstly, PSO is a stochastic evolutionary algorithm that does not incorporate survival of the fittest, and all individuals are retained as members of the population through the course of the run. Secondly, there are no conventional evolutionary operators such as crossover and mutation in PSO. Each particle adjusts its flying in the search space in terms of its own flying experience and its companions' flying experience. A detailed comparison between PSO and evolutionary optimization was shown in [22] and [23]. Successful applications of PSO to function optimization, neural network training, fuzzy system control [24], and multiobjective optimization [25] have demonstrated that PSO is a promising and efficient optimization method. Many significant improvements were proposed to improve the performance of the PSO algorithm. These amelioration and development can be placed into five categories: inertial weight varying strategy, such as constant inertial weight (CIW) [26], linearly decreasing inertial weight (LDIW) [27], fuzzy inertial weight (FIW) [28], and random inertial weight (RIW) [29]; parameter selection and convergence analysis [30]–[35]; swarm topology structure [36]–[38]; discrete PSO [39]; hybrid PSO combined with some evolutionary computation operators and other methods, such as selection [40], mutation [41], [42], breeding [43], chaos [44], cooperative approach [45], orthogonal experimental design [46], deflection and stretching techniques [47]. A detailed overview of the basic concepts of PSO and its variants was presented by del Valle *et al.* in [48].

However, all the above improvements were based on the framework of position and velocity, and did not deviate from the conventional position and velocity update rules. These proposals usually involve changes to the PSO update equations, without changing the structure of the algorithm. To this end, Sun *et al.* [49], [50] put forward a quantum-behaved particle swarm optimization (QPSO) algorithm model from the perspective of Quantum Mechanics view rather than the Newtonian rules assumed in all previous versions of PSO. QPSO was tested on some benchmark functions and experimental results showed that QPSO outperforms PSO. Mikki *et al.* [51] applied the QPSO to linear array antenna synthesis to find an equivalent circuit model for the dielectric resonator antenna (DRA). Fu *et al.* [52] proposed a 2-D path planner based on QPSO algorithm and compared the performance of QPSO with an improved version of the original PSO [17]. QPSO algorithm has been widely used in economic load dispatch [53], [54], model parameter extraction [55], multiobjective optimization of composite structures [56], multiple sequence alignment [57], QoS multicast routing [58], and many other optimization problems [59], [60]. Recently, a new version of PSO called θ-PSO was proposed in [61]. In θ-PSO, an increment of phase angle vector replaces the increment of velocity vector and the positions of particles are decided by the mapping of phase angles.

Inspired by Sun *et al.* [49], [50] and Zhong *et al.* [61], a phase angle-encoded and quantum-behaved particle swarm optimization algorithm named θ-QPSO was put forward in this paper. It was tested on several different benchmark functions to show its high performance. The primary objective of this paper is to describe a general route planning strategy based on θ-QPSO. Therefore, based on the researches in [52], the proposed θ-QPSO algorithm is employed to generate a 3-D path for UAV in the presence of different threat environments where the location and extent of each threat are known, corroborating the results obtained on the benchmark functions. In the traditional approaches [3], [6], [7], the threat cost calculation is time-consuming. To compensate for this perceived weakness, a new method for threat cost calculation was constructed and the constraints [1], [11] that must be satisfied by a qualified path were incorporated into the cost function. Using statistical method, we compare our θ-QPSO with respect to the GA, DE, PSO, θ-PSO, and QPSO through the continuous function optimization and UAV route planning-oriented application. Simulation results are given to demonstrate the effectiveness and feasibility of the proposed approach.

The remainder of this paper is organized as follows. For self-completeness, the concise description of the used PSO, θ-PSO, and QPSO is given in Section II. The proposed θ-QPSO is stated in Section III. The performance comparison of the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO on six benchmark functions is described in Section IV. Section V presents the application of the proposed θ-QPSO algorithm to the UAV route planning problem. Experimental results of route planning are shown in Section VI. Finally, the paper is concluded in Section VII.

II. PSO, Θ-PSO, AND QPSO

A. Particle Swarm Optimization

In PSO [17], each particle adjusts its position in search space according to its own flying experience and companions' flying experience. The ith particle $X_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ is associated with a velocity vector $V_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$, where d is the dimension of the problem to be solved. During the search process, every particle keeps track of the personal best $(pbest)$ position $P_i = [p_{i1}, p_{i2}, \ldots, p_{id}]$ found by itself and the global best $(gbest)$ position $P_g = [p_{g1}, p_{g2}, \ldots, p_{gd}]$ achieved by any particle in the swarm. During the iteration procedure, the velocity and position of the particle are updated according to (1) [26], [27]

$$\begin{cases} v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_{1i}^{k}\left(p_{ij}^{k} - x_{ij}^{k}\right) + c_2 r_{2i}^{k}\left(p_{gj}^{k} - x_{ij}^{k}\right) \\ x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1},\ (i = 1, 2, \ldots, m; j = 1, 2, \ldots, d) \end{cases} \tag{1}$$

where v_{ij}, whose representation is similar to x_{ij}, p_{ij}, and p_{gj}, is the jth dimension of the ith particle's velocity and usually confined to the closed interval $[-v_{\max}, v_{\max}]$ to prevent the explosion of the particles. Coefficients r_1 and r_2 are two pseudorandom scalar values drawn uniformly from the unit interval. The superscript k in (1) denotes the kth iteration. The acceleration coefficients c_1 and c_2 are 2.0 for almost all applications [27]. Factor w is the inertial weight and m is the size of swarm.

The inertial weight plays the role of balancing the global search (large inertial weight) with the local search (small inertial weight). Shi and Eberhart [30] pointed out that a time decreasing inertial weight could significantly improve the performance of the algorithm. The variation of inertial weight in PSO is analogous to that of temperature in simulated annealing. The PSO modeled in (1) is known as the Standard PSO. If the inertial weight is equal to 1, then the Standard PSO turns into the original PSO. The maximum velocity is often set to be the dynamic range of each variable on every dimension in Standard PSO, whereas about 10% to 20% of the dynamic range of each variable on every dimension in original PSO [24], [62]. It is believed that the maximum velocity $v_{\max}$ is no longer needed in the PSO with constriction factor [35], but better solutions can be found by setting the maximum velocity $v_{\max}$ to be the dynamic range of each variable on every dimension [62].

B. Phase Angle-Encoded Particle Swarm Optimization

In θ-PSO [61], the velocity vector $V_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$ and position vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ are expressed by an increment of phase angle vector $\Delta\Theta_i = [\Delta\theta_{i1}, \Delta\theta_{i2}, \ldots, \Delta\theta_{id}]$ and a phase angle vector $\Theta_i = [\theta_{i1}, \theta_{i2}, \ldots, \theta_{id}]$, respectively. Based on predefined monotone mapping, the positions of particles are obtained through mapping phase angles to the solution space. The

phase angle increment and phase angle are updated according to (2) [61]

$$\begin{cases} \Delta\theta_{ij}^{k+1} = w\Delta\theta_{ij}^{k} + c_1 r_{1i}^{k}\left(\lambda_{ij}^{k} - \theta_{ij}^{k}\right) + c_2 r_{2i}^{k}\left(\lambda_{gj}^{k} - \theta_{ij}^{k}\right) \\ \theta_{ij}^{k+1} = \theta_{ij}^{k} + \Delta\theta_{ij}^{k+1},\ (i = 1, 2, \ldots, m; j = 1, 2, \ldots, d) \\ x_{ij}^{k} = \left[(x_{\max} - x_{\min})\sin\left(\theta_{ij}^{k}\right) + x_{\max} + x_{\min}\right]/2 \end{cases} \quad (2)$$

where $\Delta\theta_{ij}$ and θ_{ij} are the jth dimension of the ith particle's phase angle increment and phase angle, both confined to the closed interval $[-\pi/2, \pi/2]$. $\Lambda_i = [\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{id}]$ and $\Lambda_g = [\lambda_{g1}, \lambda_{g2}, \ldots, \lambda_{gd}]$ are the phase angle of the ith particle's personal best $(pbest)$ position and that of the swarm's global best $(gbest)$ position. $x_{\min}$ and $x_{\max}$ are the lower and upper bounds of the search space. The rest variables in (2) are the same as those in (1).

C. Quantum-Behaved Particle Swarm Optimization

Deforming the velocity updating formula shown in (1), we have

$$v_{ij}^{k+1} = wv_{ij}^{k} + \left(c_1 r_{1i}^{k} + c_2 r_{2i}^{k}\right) \times \left[\left(\frac{c_1 r_{1i}^{k}}{c_1 r_{1i}^{k} + c_2 r_{2i}^{k}} p_{ij}^{k} + \frac{c_2 r_{2i}^{k}}{c_1 r_{1i}^{k} + c_2 r_{2i}^{k}} p_{gj}^{k}\right) - x_{ij}^{k}\right]. \quad (3)$$

Because c_1, c_2 are deterministic constants and r_1, r_2 are random numbers, (3) can be rewritten as

$$v_{ij}^{k+1} = wv_{ij}^{k} + (\xi_1 + \xi_2)\left[\left(\frac{\xi_1}{\xi_1 + \xi_2} p_{ij}^{k} + \frac{\xi_2}{\xi_1 + \xi_2} p_{gj}^{k}\right) - x_{ij}^{k}\right]. \quad (4)$$

Define $\xi = \xi_1 + \xi_2$ and $\xi_1/(\xi_1 + \xi_2) = \zeta$, then

$$v_{ij}^{k+1} = wv_{ij}^{k} + \xi\left(q_{ij}^{k} - x_{ij}^{k}\right),\ q_{ij}^{k} = \zeta p_{ij}^{k} + (1 - \zeta)p_{gj}^{k}. \quad (5)$$

Equations (3)–(5) are mathematically identical to (1). Clerc and Kennedy [35] have illustrated that as the system iterates, each particle is pulled toward its local attractor $Q_i = [q_{i1}, q_{i2}, \ldots, q_{id}]$ based on the interaction of particles' individual searches and group's public search to ensure convergence.

QPSO algorithm [49] assumes that there is 1-D delta potential well on each dimension at the local attractor point, and every particle in the swarm has quantum behavior. The quantum state of a particle is depicted by a wave function instead of the position and velocity used in PSO. The probability density function of the position that each particle flies to can be deduced from the Schrödinger equation. The measurement of the position for each particle from the quantum state to the classical one can be implemented by using Monte Carlo simulation method, and the position of the ith particle is updated as follows:

$$x_{ij}^{k+1} = q_{ij}^{k} \pm \left(l_{ij}^{k}/2\right)\ln(1/u) \quad (6)$$

where u is a uniformly distributed pseudorandom number on the unit interval. q_{ij} and l_{ij} are the jth dimension of the ith particle's local attractor and delta potential well characteristic length, defined as follows [50]:

$$l_{ij}^{k} = 2b\left|mbest_{j}^{k} - x_{ij}^{k}\right|,\ mbest_{j}^{k} = \sum_{i=1}^{m} p_{ij}^{k}/m \quad (7)$$

where b is the contraction-expansion coefficient and $mbest$ is the mean best position of the population. Substituting (5) and (7) into (6), the position of particle can be updated as follows:

$$x_{ij}^{k+1} = \zeta p_{ij}^{k} + (1 - \zeta)p_{gj}^{k} \pm b\left|\sum_{i=1}^{m} p_{ij}^{k} \Big/ m - x_{ij}^{k}\right|\ln(1/u). \quad (8)$$

III. Phase Angle-Encoded and Quantum-Behaved Particle Swarm Optimization Algorithm

Based on the aforementioned θ-PSO and QPSO, we proposed the phase angle-encoded and quantum-behaved particle swarm optimization algorithm, called θ-QPSO. Just like the θ-PSO, the position vector X_i is replaced by a phase angle vector $\Theta_i = [\theta_{i1}, \theta_{i2}, \ldots, \theta_{id}]$, but an increment of phase angle vector $\Delta\Theta_i = [\Delta\theta_{i1}, \Delta\theta_{i2}, \ldots, \Delta\theta_{id}]$ is not necessary in θ-QPSO, which is fundamentally distinct from θ-PSO. This radical difference is derived from the fact that the particles in PSO and QPSO are manipulated in different ways [52]. Namely, the θ-QPSO only maintains the information regarding phase angle (equivalent to position), whereas the phase angle (equivalent to position) and its increment (equivalent to velocity) both need to be retained in θ-PSO, resulting in expensive computation and a large amount of memory overhead.

Let $f : [-\pi/2, \pi/2] \to [x_{\min}, x_{\max}]$ be a monotone mapping, then there is one and only one position in the solution space corresponding to any given phase angle, and vice versa. In fact, because of the one-to-one correspondence of f, the inverse mapping of f exists. Denote $f^{-1} : [x_{\min}, x_{\max}] \to [-\pi/2, \pi/2]$ as the inverse mapping of f, then there is one and only one phase angle corresponding to any given position in the solution space. Consider a population of m particles, the phase angle of each particle is updated according to

$$\begin{cases} \gamma_{ij}^{k} = \mu\lambda_{ij}^{k} + (1 - \mu)\lambda_{gj}^{k},\ \psi_{j}^{k} = \sum_{i=1}^{m} \lambda_{ij}^{k}/m \\ \theta_{ij}^{k+1} = \gamma_{ij}^{k} \pm b\left|\psi_{j}^{k} - \theta_{ij}^{k}\right|\ln(1/u) \\ x_{ij}^{k} = f\left(\theta_{ij}^{k}\right), (i = 1, 2, \ldots, m; j = 1, 2, \ldots, d) \end{cases} \quad (9)$$

where μ is a uniform random variable within [0, 1]. The phase angle of the ith particle's local attractor $\Gamma_i = [\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{id}]$ is defined as the stochastic weighted mean of $\Lambda_i = [\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{id}]$ and $\Lambda_g = [\lambda_{g1}, \lambda_{g2}, \ldots, \lambda_{gd}]$. Consequently, Γ_i is a random position located within the hypercube spanned by Λ_i and Λ_g. The corners of the hypercube Λ_i and Λ_g are the phase angle of the ith particle's personal best $(pbest)$ position and phase angle of the swarm's global best $(gbest)$ position. The phase angle of the swarm's mean best position $\Psi = [\psi_1, \psi_2, \ldots, \psi_d]$ is defined as the arithmetical average of $[\Lambda_1, \Lambda_2, \ldots, \Lambda_m]^{\mathrm{T}}$. The rest variables in (9) are the same as those in (1), (2), and (8).

Suppose F is the fitness function of the problem to be minimized at hand, then the phase angle Λ_i and Λ_g can be computed using

$$\Lambda_i^{k+1} = \arg\min_{1 \le j \le k+1} F\left[f\left(\Theta_i^{j}\right)\right] = \begin{cases} \Lambda_i^{k}, & \text{if } F\left[f\left(\Lambda_i^{k}\right)\right] \le F\left[f\left(\Theta_i^{k+1}\right)\right] \\ \Theta_i^{k+1}, & \text{if } F\left[f\left(\Lambda_i^{k}\right)\right] > F\left[f\left(\Theta_i^{k+1}\right)\right] \end{cases} \quad (10)$$

$$\Lambda_g^{k+1} = \arg\min_{1 \le i \le m} F\left[f\left(\Lambda_i^{k+1}\right)\right] = \begin{cases} \Lambda_g^{k}, & \text{if } F\left[f\left(\Lambda_g^{k}\right)\right] \le F\left[f\left(\Lambda_i^{k+1}\right)\right] \\ \Lambda_i^{k+1}, & \text{if } F\left[f\left(\Lambda_g^{k}\right)\right] > F\left[f\left(\Lambda_i^{k+1}\right)\right]. \end{cases} \quad (11)$$

As with the θ-PSO, the phase angle update equation of θ-QPSO is composed of a cognition term and a social one. In the phase angle update equation of θ-QPSO, the term $\mu\lambda_{ij}$ is associated with cognition since it only takes into account the particle's own experience, while the term $(1 - \mu)\lambda_{gj} \pm b|\psi_j - \theta_{ij}|\ln(1/u)$ represents the social interaction between particles. Moreover, the introduction of mean best phase angle Ψ results in more cooperation and facilitates the information sharing and communication between particles. Theoretically, the performance of θ-QPSO can be improved relatively compared to θ-PSO.

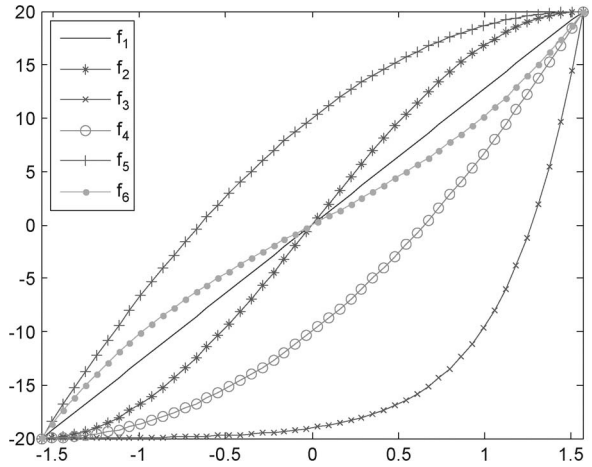

Fig. 1. Comparison of different f.

Because $f : [-\pi/2, \pi/2] \rightarrow [x_{\min}, x_{\max}]$ is the mapping from phase angle to the position in search space, the selection of f is of great importance. To find out the impact of f on the algorithm, six different choices of f are studied, including the f_2 used in [61], resulting in six versions of θ-QPSO algorithms from θ-QPSO1 to θ-QPSO6. Plots of $f_1 - f_6$ are illustrated in Fig. 1 by setting $[x_{\min}, x_{\max}] = [-20, 20]$.

f_1, a linear and monotonic increasing function

$$f_1(\theta) = (x_{\max} - x_{\min})\theta/\pi + (x_{\max} + x_{\min})/2. \quad (12)$$

f_2, a sinusoidal function, which is convex in interval $[-\pi/2, 0]$ and concave in interval $[0, \pi/2]$

$$f_2(\theta) = [(x_{\max} - x_{\min})\sin(\theta) + x_{\max} + x_{\min}]/2. \quad (13)$$

f_3, a monotonic increasing exponential function

$$f_3(\theta) = \left[(x_{\max} - x_{\min} + \sqrt{(x_{\max} - x_{\min})^2 + 4})/2\right]^{2\theta/\pi} + (x_{\max} + x_{\min} - \sqrt{(x_{\max} - x_{\min})^2 + 4})/2. \quad (14)$$

f_4, a quadratic and convex function in $[-\pi/2, \pi/2]$

$$f_4(\theta) = (x_{\max} - x_{\min})(\theta/\pi + 1/2)^2 + x_{\min}. \quad (15)$$

f_5, a quadratic and concave function in $[-\pi/2, \pi/2]$

$$f_5(\theta) = -(x_{\max} - x_{\min})(\theta/\pi + 1/2)^2 + 2(x_{\max} - x_{\min})(\theta/\pi + 1/2) + x_{\min}. \quad (16)$$

f_6, a hyperbolic sine function, which is concave in interval $[-\pi/2, 0]$ and convex in interval $[0, \pi/2]$

$$f_6(\theta) = [(x_{\max} - x_{\min})\sinh(\theta)/\sinh(\pi/2) + x_{\max} + x_{\min}]/2. \quad (17)$$

The detailed implementation of the θ-QPSO algorithm can be described as follows:

Step 1) Initialize the phase angle of each particle randomly in the interval $[-\pi/2, \pi/2]$.

TABLE I
BENCHMARK FUNCTIONS

Function	Analysis Formula	Domain
Sphere	$F_1(X_i) = \sum_{j=1}^{d} x_{ij}^2$	[-15,15]
Quadric	$F_2(X_i) = \sum_{k=1}^{d} (\sum_{j=1}^{k} x_{ij})^2$	[-100,100]
Ackley	$F_3(X_i) = 20 + e - 20\exp(-0.2\sqrt{\sum_{j=1}^{d} x_{ij}^2 / d}) - \exp(\sum_{j=1}^{d} \cos(2\pi x_{ij})/d)$	[-30,30]
Rosenbrock	$F_4(X_i) = \sum_{j=1}^{d-1} [100(x_{i,j+1} - x_{ij}^2)^2 + (x_{ij} - 1)^2]$	[-30,30]
Rastrigrin	$F_5(X_i) = \sum_{j=1}^{d} [x_{ij}^2 - 10\cos(2\pi x_{ij}) + 10]$	[-5.12,5.12]
Griewank	$F_6(X_i) = \sum_{j=1}^{d} x_{ij}^2 / 4000 - \prod_{j=1}^{d} \cos(x_{ij}/\sqrt{j}) + 1$	[-600,600]

Step 2) Choose one f from $f_1 - f_6$ defined in (12)–(17) to calculate the position corresponding to phase angle.

Step 3) Evaluate each particle based on the user-specified fitness function F.

Step 4) Compute the phase angle of each particle's personal best $(pbest)$ position and phase angle of the swarm's global best $(gbest)$ position using (10) and (11).

Step 5) Compute the phase angle of each particle's local attractor and the phase angle of the swarm's mean best $(mbest)$ position from (9).

Step 6) Renew the phase angle of each particle in the swarm according to (9) and confine them to $[-\pi/2, \pi/2]$.

Step 7) Repeat Steps 2) to 6) until the stopping condition is met.

The stopping criterion depends on the type of problem being solved. Usually, the algorithm is run for a fixed number of objective function evaluations (thus a fixed number of iterations) and ends when the process starting with Step 2) has been repeated a predefined number of times. Another termination condition available is a pre-specified error bound. In this case, the algorithm is run for the user-defined maximum number of iterations, but at any time if the pre-specified error threshold is reached, then the algorithm stops. The theoretical optimal value of the problem to be optimized is needed to be known beforehand when the second criterion is adopted, whereas the theoretical optimal values of many real-world engineering optimization problems are unknown in advance.

IV. Θ-QPSO FOR FUNCTION OPTIMIZATION

Many popular optimization algorithms are deterministic, like the gradient-based algorithms. The PSO-based algorithms, similar to the algorithms belonging to the Evolutionary Algorithm family, are stochastic algorithms that do not need gradient information derived from the error function. This allows the PSO-based algorithms to be used on functions where the gradient is either unavailable or computationally expensive to obtain.

To compare the performance of different versions of θ-QPSO algorithms for continuous function optimization, six well-known benchmark functions are selected for testing, and a total of 50 runs are performed on each test function. The function name, analytic expression, and variable domain for each function are listed in Table I.

The global minimum for each function is 0 and the corresponding global minimizer is $(0, 0, \ldots, 0)$ except Rosenbrock function, whose minimizer is $(1, 1, \ldots, 1)$. The Sphere, Quadric, and Rosenbrock are three unimodal functions, and the rest functions are multimodal. The Rosenbrock function has a narrow valley from the perceived local

TABLE II
NUMERICAL RESULTSOF THE SIX Θ-QPSO ALGORITHMS ON THE SIX BENCHMARK FUNCTIONS

Algorithm	F_1		F_2		F_3		F_4		F_5		F_6	
	Mean	St Dev	Mean	St Dev	Mean	St Dev	Mean	St Dev	Mean	St Dev	Mean	St Dev
θ-QPSO1	4.53e-18	1.01e-17	3.40e-25	7.61e-25	8.88e-16	1.67e-16	6.01e+1	1.15e+2	1.00e-3	0.00e+0	1.00e-3	0.00e+0
θ-QPSO2	5.24e-45	1.17e-44	4.58e-36	1.02e-35	1.59e-15	1.58e-15	8.66e+0	4.71e-1	1.00e-3	0.00e+0	1.00e-3	0.00e+0
θ-QPSO3	4.20e+1	1.97e+1	7.19e+3	3.48e+3	1.27e+1	1.17e+0	1.47e+6	1.50e+6	5.25e+1	8.89e+0	1.45e+1	4.19e+0
θ-QPSO4	1.19e+1	4.17e+0	3.21e+3	5.80e+2	9.00e+0	2.56e+0	1.28e+5	1.16e+5	4.08e+1	1.34e+1	6.15e+0	1.41e+0
θ-QPSO5	1.91e+1	1.11e+1	3.88e+3	1.58e+3	9.86e+0	1.41e+0	5.07e+5	7.15e+5	4.54e+1	1.29e+1	1.17e+1	7.49e+0
θ-QPSO6	5.45e-24	1.21e-32	1.50e-5	3.35e-5	2.54e-6	5.69e-6	1.42e+2	1.91e+2	1.80e+1	2.54e+1	1.00e-3	0.00e+0

optima to the global optimum. The Girewank function has a significant interaction between its variables due to the product term. The fitness values of Sphere, Quadric, Ackley, and Rosenbrock are defined as their function values directly while the fitness values of Rastrigrin and Griewank are defined as their function values plus 10^{-3}.

The initial positions of particles are randomly generated from the uniform distribution on the intervals given in Table I. Following the suggestion of Sun *et al.* [49], [50], the contraction-expansion coefficient is decreased linearly from 1.0 to 0.5. The population size m and dimension d defined in (1) are set to 20 and 10, respectively.

For fair comparison, six θ-QPSO algorithms use the same pre-specified number of function evaluations as the stopping condition. All experiments were run for 10 000 objective function evaluations, which is equivalent to 500 iterations using a swarm size of 20. The average best fitness values (Mean) of the 50 trials along with its standard deviation (St Dev) for each combination of function and algorithm are shown in Table II. The convergence curves of the six θ-QPSO algorithms on the six benchmark functions are plotted in Fig. 2(a)–(f), respectively. Each convergence curve shows the mean best fitness values of 50 runs per iteration. Suppose $F(k,l)$ to be the best fitness value at the kth generation of the lth run, then $Mean\_F(k) = [F(k,1) + \cdots + F(k,num\_run)]/num\_run$ is the mean best fitness value of num_run runs at the kth iteration, where num_run is the predefined number of runs.

We compare the performance of all algorithms based on the following three factors: solution quality, stability, and convergence speed of each algorithm. The solution quality can be expressed by the mean value of best fitness, also accounting for the search ability of an algorithm. Therefore, for minimization problem, the smaller the mean value of best fitness, the higher the quality of solution, and the stronger the search ability. Similarly, the stability of an algorithm can be reflected by the standard deviation. The larger the standard deviation value, the worse the stability of an algorithm.

The convergence speed of an algorithm is defined as how fast the algorithm can converge to the optimal or sub-optimal solution. Hence, under the given maximum number of iterations, the smaller the number of iterations needed to ensure that the algorithm converges to the optimal or sub-optimal solution, the faster the convergence speed. For example, as shown in Fig. 2(e), the θ-QPSO2 converges faster than θ-QPSO1 on Rastrigrin since the former needs about 200 iterations to converge to 10^{-3}, while the latter requires about 300 iterations to converge to the same level.

Alternatively, the convergence curve is also used to intuitively show the convergence rate of an algorithm. It plots the average best fitness value at each generation. If the mean best fitness of algorithm A is always better than that of algorithm B at every generation, the convergence speed of algorithm A is strictly faster than that of algorithm B. In other words, if algorithm A converges strictly faster than algorithm B, we can draw the following two conclusions:

1) For any $F \geq F^*$, we have $k_A \leq k_B$, where F^* is the fitness of optimal solution, while k_A and k_B are the minimum number of iterations needed by algorithm A and algorithm B to converge to F, respectively;
2) For any $k \in \{1, 2, \ldots, max\_iter\}$, we have $Mean\_F_A(k) < Mean\_F_B(k)$, where $Mean\_F_A(k)$ is the mean best fitness of algorithm A at the kth iteration and max_iter is the predefined maximum number of iterations.

Note that both 1) and 2) can be regarded as the sufficient condition that algorithm A converges faster than algorithm B. According to the above analysis, it is easy to determine which algorithm converges faster based on the position relationship between their convergence curves. For minimization problem, if the convergence curve of algorithm A is always below that of algorithm B, then the convergence rate of algorithm A is strictly faster than that of algorithm B. However, this is not always the case. The convergence curve of algorithm A may be below that of algorithm B at the early stage, and then above that of algorithm B gradually, such as the convergence curve of θ-QPSO2 and that of θ-QPSO1 shown in Fig. 2(c). The θ-QPSO2 converges faster than θ-QPSO1 from the beginning to about the 150th iteration, and slower than θ-QPSO1 from the 150th iteration to the 300th iteration. Therefore, the convergence speed of θ-QPSO1 and that of θ-QPSO2 on Ackley are nip and tuck.

Table II reveals that the θ-QPSO2 has the best performance on all six benchmark functions in terms of the above analysis. By looking at the convergence curves shown in Fig. 2, it is not difficult to find that the θ-QPSO1 performs almost as well as the θ-QPSO2 on Rastrigrin and Griewank although the convergence speed of the θ-QPSO1 is slower. Moreover, the θ-QPSO1 achieves better results than the θ-QPSO2 on Ackley, but falls far behind the θ-QPSO2 on Sphere, Quadric, and Rosenbrock. It is interesting that the θ-QPSO6 also can find the global optimum of Griewank, and even converges faster than the θ-QPSO2. However, the performance of the θ-QPSO6 on the rest five functions is inferior to that of the θ-QPSO2.

The performance of the θ-QPSO3, θ-QPSO4, and θ-QPSO5 on the six test functions are all poor and far from perfect, which can be explained from shapes of f_3, f_4, and f_5 as illustrated in Fig. 1. Such mappings have different shapes, but all share two characteristics: one is that they are nonlinear function different from f_1, the other is that they hold the convexity or concavity of themselves in the whole domain of variability different from f_2 and f_6. The high performance of θ-QPSO2 arises mainly from the effect of mapping f_2.

These experimental results showed that the mapping f_2 defined in (13) is the best choice among $f_1 - f_6$ for θ-PSO and θ-QPSO to optimize the benchmark functions given in Table I. To further compare the performance of the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO for continuous function optimization, the six benchmark functions listed in Table I are selected for testing, and a total of 50 runs are performed on each test function. The crossover and mutation probability for the used real-valued GA [13], [14] are 0.7 and 0.05, respectively. Roulette wheel selection, arithmetic crossover, and random mutation are used as genetic operators. The mutation factor and crossover constant of DE [16], [18] are 0.9 and 0.85, respectively. Following the

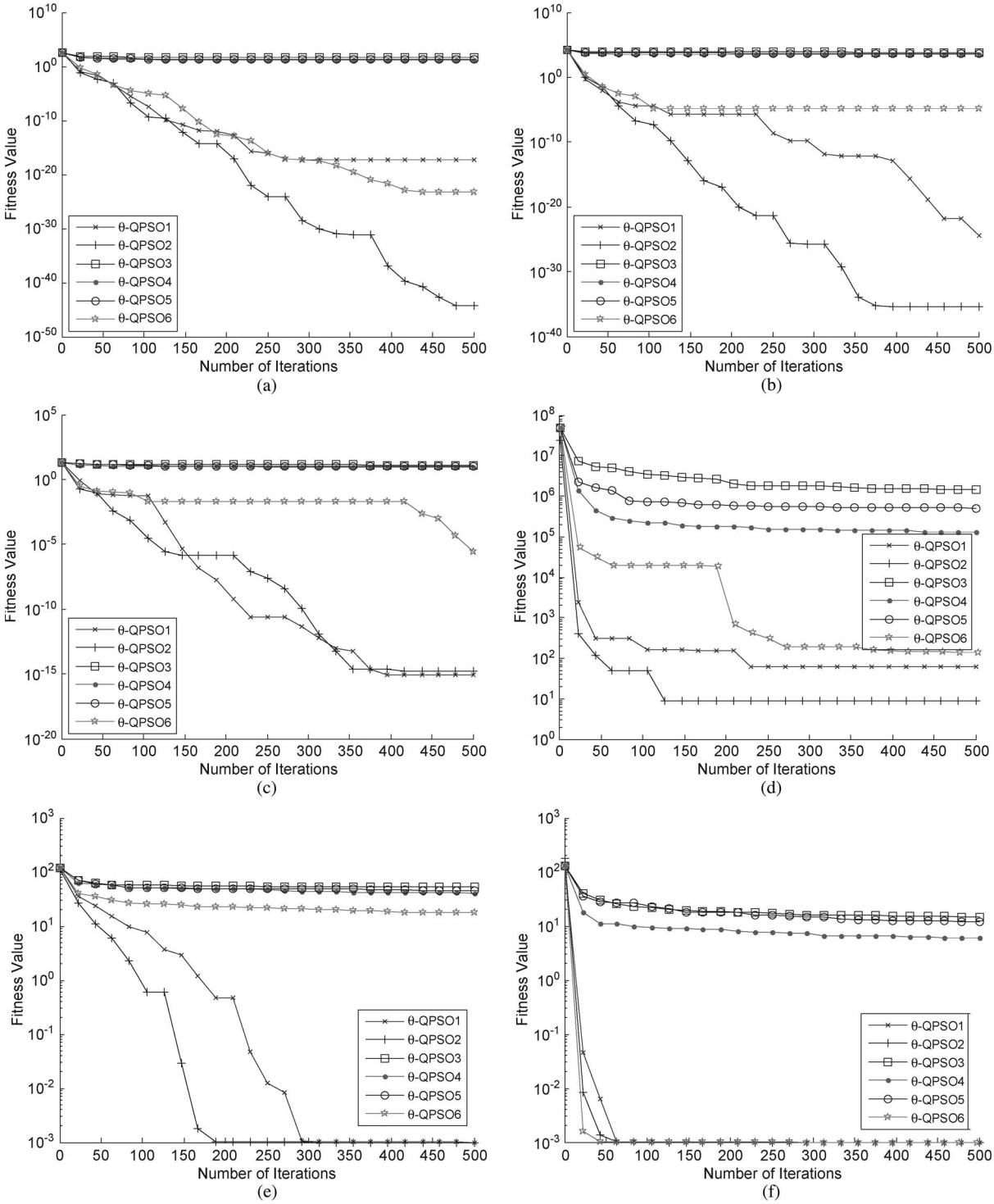

Fig. 2. Convergence curves of the six θ-QPSO algorithms on the six benchmark functions in (a)–(f), respectively: (a) Sphere function; (b) Quadric function; (c) Ackley function; (d) Rosenbrock function; (e) Rastrigrin function; (f) Griewank function.

suggestion of Shi and Eberhart [27], the inertial weight coefficient w was varied linearly from 0.9 to 0.4, and the $v_{\max}$ used in PSO was set to $x_{\max}$. Other parameter settings of the system used the same settings of θ-QPSO2. In other words, the parameters of θ-QPSO2 are used for the proposed θ-QPSO. The above parameters values are selected based on the suggestions in other literature where these values have been found, empirically, to provide good performance [13], [14], [16], [27], [49], [50]. The average best fitness values of the 50 trials along with its standard deviation for each combination of function and algorithm are recorded in Table III. The convergence curves of the GA, DE, and four PSO-based algorithms on the six benchmark functions are shown in Fig. 3(a)–(f), respectively. Each convergence curve shows the mean best fitness values of 50 runs per iteration.

From the results, it is observed that θ-QPSO achieves better results on all six benchmark functions than the GA, DE, and three other PSO-based algorithms. It also can be seen from Table III that θ-QPSO outperforms the PSO, θ-PSO, and QPSO considerably on Sphere,

TABLE III
NUMERICAL RESULTS OF THE GA, DE, AND FOUR PSO-BASED ALGORITHMS ON THE SIX BENCHMARK FUNCTIONS

Algorithm	F_1		F_2		F_3		F_4		F_5		F_6	
	Mean	St Dev	Mean	St Dev	Mean	St Dev	Mean	St Dev	Mean	St Dev	Mean	St Dev
GA	4.29e+0	5.11e+0	1.54e+3	5.43e+2	9.22e+0	3.02e+0	1.69e+3	1.51e+3	2.57e+1	9.11e+0	2.01e+0	3.81e-1
DE	3.26e-2	2.87e-2	6.93e+1	6.21e+1	7.71e-1	8.84e-1	1.55e+2	2.06e+2	6.92e+0	5.74e+0	7.24e-1	1.09e-1
PSO	8.34e-4	7.34e-4	7.25e+0	1.25e+1	1.01e+0	1.02e+0	1.11e+2	2.03e+2	7.32e+0	6.86e+0	1.93e-1	2.12e-1
θ-PSO	7.94e-18	1.77e-17	1.94e-15	4.33e-15	1.29e-11	1.56e-11	8.89e+0	1.34e-1	1.99e-3	2.20e-3	1.00e-3	0.00e+0
QPSO	3.49e-26	6.51e-26	1.84e-29	4.11e-29	5.07e-12	1.03e-11	2.27e+1	3.65e+1	3.44e+0	3.77e+0	6.52e-2	8.40e-2
θ-QPSO	5.24e-45	1.17e-44	4.58e-36	1.02e-35	1.59e-15	1.58e-15	8.66e+0	4.71e-1	1.00e-3	0.00e+0	1.00e-3	0.00e+0

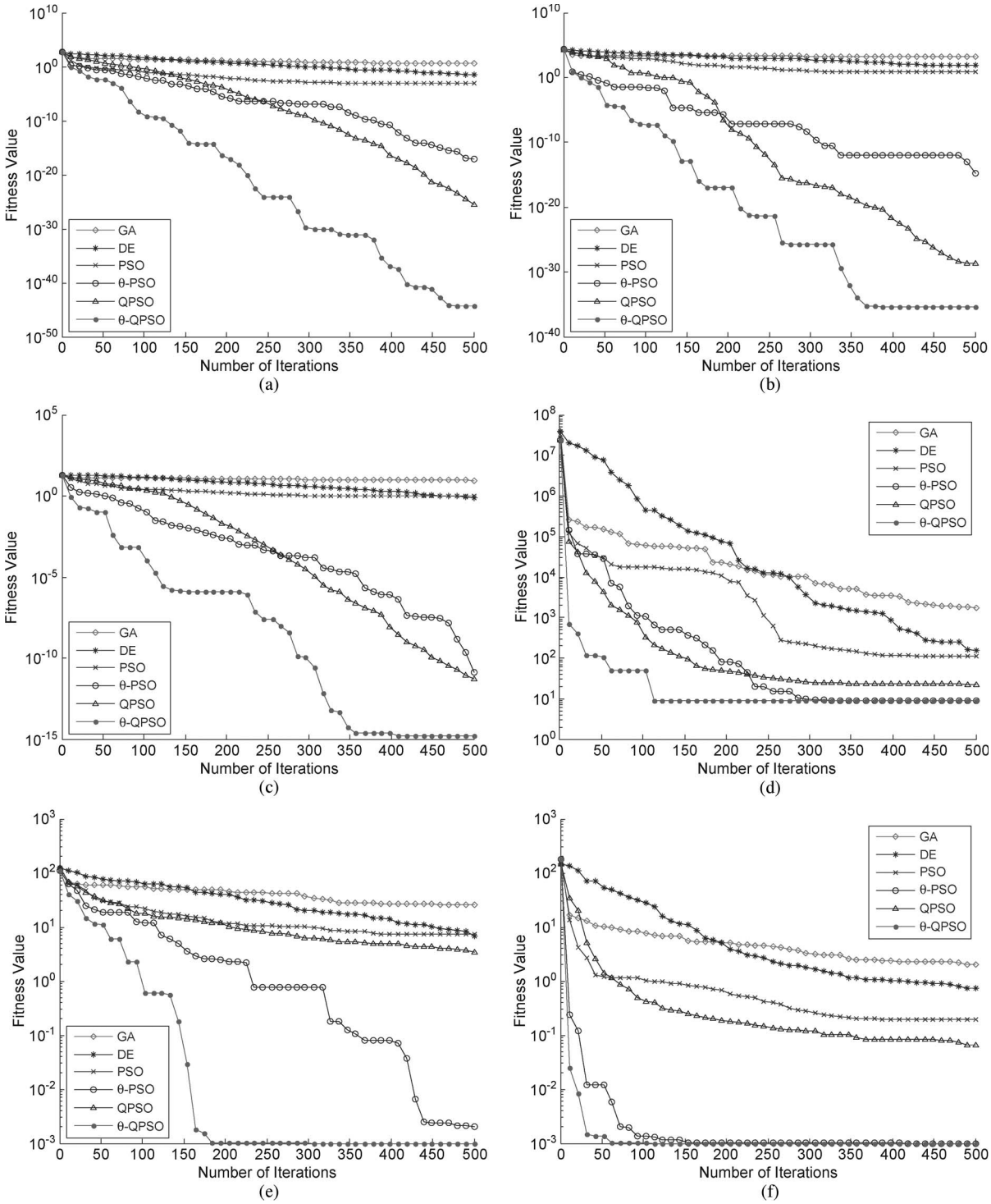

Fig. 3. Convergence curves of the GA, DE, and four PSO-based algorithms on the six benchmark functions in (a)–(f), respectively: (a) Sphere function; (b) Quadric function; (c) Ackley function; (d) Rosenbrock function; (e) Rastrigrin function; (f) Griewank function.

Quadric, and Ackley. The average best fitness value produced by θ-QPSO on Sphere drops down by 41, 27, and 19 orders of magnitude compared with that resulted by the PSO, θ-PSO, and QPSO, respectively. It should be noted that θ-QPSO can find the global minimum of Rastrigrin and Griewank with zero standard deviation. This means that the global minimum of these two functions can be reached by θ-QPSO during every run and it has good stability. Through comparing the convergence curves shown in Fig. 3, it is easy to find that θ-QPSO converges faster than the GA, DE, and three other PSO-based algorithms on all six benchmark functions. Among the six used algorithms, the performance of GA is the poorest, regardless of whether the evaluation criterion is the solution quality, robustness, or convergence speed. The DE achieves better results than PSO on Ackley and Rastrigrin. However, the PSO has faster convergence speed than DE on all test functions. The advantages of θ-PSO and θ-QPSO over PSO and QPSO are remarkable on Rastrigrin and Griewank as shown in Fig. 3. Both θ-PSO and θ-QPSO can not only find higher quality solutions but also possess faster convergence speed than the GA, DE, PSO, and QPSO on most multimodal functions. Like θ-QPSO, θ-PSO also can hit the global minimum of Griewank with the standard deviation of 0, but offers a slower convergence rate in contrast with θ-QPSO. The θ-PSO also achieves similar results as the θ-QPSO on Rosenbrock, and they are both better than the GA, DE, PSO, and QPSO on this function. Although QPSO surpasses PSO and θ-PSO on the first three functions, its performance on the rest three functions is inferior to θ-PSO and θ-QPSO remarkably.

V. Θ-QPSO for UAV Route Planning

A. Route Representation

Each possible solution in the problem space can be represented by an individual of the population. An individual in swarm is referred to as a particle. In fact, a particle with arbitrarily small mass and volume is a potential path in the search space. The entries of each particle represent the coordinates of path nodes. Consequently, the particle representation builds a one-to-one mapping between particles and candidate paths. The terms "particle," "individual," "path," and "solution" are used interchangeably in this paper.

The phase angle-encoded population is denoted by a matrix $\Theta = [\Theta_1, \Theta_2, \ldots, \Theta_m]^{\mathrm{T}}$, where m is the size of the population. The start point and endpoint of a path do not engage in the coding of particles since all particles have the same start point and endpoint. Suppose each path is composed of n path nodes except the first node (start point) and the last one (endpoint), then the total number of path nodes contained in a potential path is $n+2$. To record the spatial position (x, y, z) of each path node, the dimension of each particle should be $3n$. As a result, each path is represented by the following fixed-length phase angle-encoded vector:

$$\Theta_i = [\theta_{i1}, \ldots, \theta_{in}, \theta_{i,n+1}, \ldots, \theta_{i,2n}, \theta_{i,2n+1}, \ldots, \theta_{i,3n}]. \quad (18)$$

Using the mapping f, we can get the position vector of each particle

$$X_i = f(\Theta_i) = [x_{i1}, \ldots, x_{in}, x_{i,n+1}, \ldots, x_{i,2n}, x_{i,2n+1}, \ldots, x_{i,3n}] \quad (19)$$

where $x_{ij} = f(\theta_{ij})$ is the jth dimension of the ith particle's position $(i = 1, \ldots, m; j = 1, \ldots, 3n)$. The first one third of X_i from x_{i1} to x_{in} and the second one third of X_i from $x_{i,n+1}$ to $x_{i,2n}$ represent the abscissa values and ordinate values of the n path nodes on the ith path, while the rest of X_i from $x_{i,2n+1}$ to $x_{i,3n}$ represent the corresponding height values of the n path nodes on the ith path. Thus, the jth path node on the ith path is specified by the 3-D coordinates $(x_{ij}, x_{i,j+n}, x_{i,j+2n}), j = 1, \ldots, n$. Suppose W_{ij} represents the jth waypoint on the ith path, then X_i can be rewritten as

$$X_i = \{W_{i1}, W_{i2}, \ldots, W_{in}\}, i = 1, \ldots, m. \quad (20)$$

B. Search Space and Initialization

Similar to the classical function optimization as studied in Section IV, reasonable ranges of parameters to be optimized should be given. Namely, after the parameters to be minimized or maximized are chosen, the next step will be defining the solution space. This requires the specification of a minimum and maximum value for each dimension of a particle. For our problem, $x_{ij} (i = 1, \ldots, m; j = 1, \ldots, 3n)$ just are the parameters that need to be optimized, and their domains are given by

$$\begin{cases} 0 \le x_{ij} \le x_{\max}, j = 1, \ldots, n \\ 0 \le x_{ij} \le y_{\max}, j = n+1, \ldots, 2n \\ z_{\min} + h \le x_{ij} \le z_{\max} + H, j = 2n+1, \ldots, 3n \end{cases} \quad (21)$$

where $x_{\max}$ and $y_{\max}$ are the horizontal and vertical components of the UAV's maximum route distance, respectively, (also can be set to be the length and width of the battlefield). h and H are the minimum and maximum elevation of the war field terrain. The boundary $z_{\min}$ and $z_{\max}$ are the minimum and maximum relative flight altitude of UAV. In other words, $z_{\min}$ and $z_{\max}$ are the minimum and maximum safe clearance between UAV and terrain. In fact, on one hand, UAV should fly at a low altitude so as to not be detected by a radar and enhance the terrain masking effect; on the other hand, the lower the flight height, the larger the probability of crash. Therefore, the solution space of our problem is defined as a cuboid, whose length, width, and height are $x_{\max}, y_{\max}$, and $(z_{\max} + H - z_{\min} - h)$, respectively.

At the beginning of system running, each particle is initialized randomly in the interval $[-\pi/2, \pi/2]$ although some more sophisticated initialization strategies (such as Sobol sequence generator, nonlinear simplex method, and skeletonization approach) can improve the overall performance of algorithm. The following mapping, a variation of f_2, can guarantee the position corresponding to each phase angle within the bounds of the search space

$$f(\theta_{ij}) = \begin{cases} x_{\max} (\sin(\theta_{ij}) + 1)/2, j \in [1, n] \\ y_{\max} (\sin(\theta_{ij}) + 1)/2, j \in [n+1, 2n] \\ ((z_{\max} + H - z_{\min} - h) \sin(\theta_{ij}) \\ \quad + z_{\max} + H + z_{\min} + h)/2, j \in [2n+1, 3n]. \end{cases} \quad (22)$$

The population size is a parameter specified by user. Theoretically, the increase of population size leads to more adequate exploration of the search space at the expense of more objective function evaluations and computation time. Shi and Eberhart [27] have found that PSO-based algorithms are not sensitive to the population size. The length of a particle, i.e., the dimension, which is equal for each particle, is another problem-dependent parameter determined by user. Note that the value of dimension makes a tradeoff between accuracy of the path and computational efficiency. In fact, as shown in (18) and (19), the dimension of a particle is three times as large as the number of nodes of a candidate path. Therefore, more path nodes may lead to more accurate path but result in larger memory overhead and more intensive computation.

C. Model of Evaluation Function

Designing reasonable fitness function is of great importance when an optimization algorithm is applied to solve actual problems. The performance of each particle is measured by the fitness function, which in general is associated with a specific application and provides the inextricable link between the optimization algorithm and physical world. Consequently, the core is to set up accurate and effective representation model of cost function for a route planning application. Once the model of cost function is established, it becomes the sole criterion for evaluating whether a particle or path is good or not. The smaller the cost value, the better the path.

Cost function used in available literature [3], [6], [7], [11] is normally composed of at least the length cost and threat cost of path, since the constraints that have to be satisfied by a qualified path should include but not be limited to: 1) minimize the path length to the target; 2) minimize the UAV' exposure to threats. Unfortunately, these two primary minimizations are impossible to be met simultaneously. We can obtain a shorter path with less regard of the exposure to the threats, such as radar or surface-to-air missile (SAM), while gain a threat-avoidance path with longer path length. Therefore, a suitable selection of the weight coefficient provides a tradeoff between path length and threat avoidance. Aside from the length cost and threat cost of path, other constraints that have to be satisfied by the UAV such as the restrictions of turning angle α, climbing/diving angle β, and flying height h, etc. are needed to be incorporated into the cost function [52].

To combine the aforementioned five types of costs, the cost function for a specific path X_i is weighted by the path length, threat, turn angle, climb/dive angle, and height

$$J(X_i) = \sum_{k=1}^{5} w_k J_k(X_i) \tag{23}$$

where $J_1(X_i) - J_5(X_i)$ are the costs of path length, threat, turn angle, climb/dive angle, and height, respectively. $w_1 - w_5$ are the corresponding weight coefficients of $J_1(X_i) - J_5(X_i)$.

For a given path X_i, the coordinates of the path nodes can be represented as $\{(x_{i1}, x_{i,n+1}, x_{i,2n+1}), (x_{i2}, x_{i,n+2}, x_{i,2n+2}), \ldots, (x_{in}, x_{i,2n}, x_{i,3n})\}$ or $\{W_{i1}, W_{i2}, \ldots, W_{in}\}$. The path length cost J_1 is defined as the sum of all path segment lengths from the start point to endpoint

$$J_1(X_i) = \sum_{j=0}^{n} \left\| \overrightarrow{W_{ij}W_{i,j+1}} \right\| \tag{24}$$

where W_{i0} and $W_{i,n+1}$ are the start point and endpoint, respectively. Note that both W_{i0} and $W_{i,n+1}$ keep unchanged for all $i = 1, \ldots, m$, since all particles have the same start point and endpoint. The symbol $\|\cdot\|$ represents the Euclidean distance of a vector (the same hereinafter).

The threat cost J_2 is computed by using the following rules. Without loss of generality, K threat sets $\{T_1, T_2, \ldots, T_K\}$ are represented by circles with different center point C_k and radius R_k. The center records the position of a threat, and length of radius indicates the cover range of a threat. For a given path segment $\overrightarrow{W_{ij}W_{i,j+1}}$, Fig. 4(b) graphically shows how to calculate the threat cost of a path segment $\overrightarrow{W_{ij}W_{i,j+1}}$ using the traditional approach. This conventional method first samples $\overrightarrow{W'_{ij}W'_{i,j+1}}$, which is the projection of path segment $\overrightarrow{W_{ij}W_{i,j+1}}$ in xoy plane, then computes the threat cost between each sampling spot and each threat center whether it is inside the threat circle or outside the threat circle, resulting in a tricky problem. On one hand, high sampling rate will cost a large amount of time to calculate the threat cost. On the other hand, the threat cost is not accurate enough and cannot represent the real threat cost when sampling rate is too low.

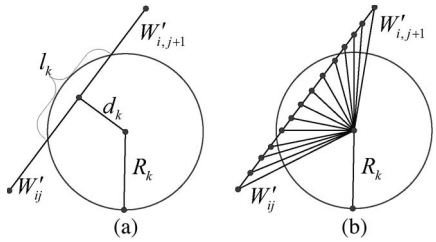

Fig. 4. Threat cost computation: (a) new method of threat cost computation; (b) traditional method of threat cost computation.

In consideration of the conflict between sampling accuracy and computational efficiency, a new method based on the location relationship between path segment and threat circle is proposed in this paper, as shown in Fig. 4(a). In our approach, the path segments are classified into two types: intersecting with threat circle and not intersecting with threat circle (including tangency and remoteness). There is no need to calculate the threat cost of those path segments that do not intersect with any threat circle. In other words, UAV is allowed to intersect a point on the boundary of a threat but with high threat cost to fly through the interior of a threat. The threat cost of a path segment that intersects one of the threat circles is proportional to the length of path segment that is contained in that threat circle.

The detailed implementation of the new method of threat cost computation can be described as follows:

Step 1) For each threat T_k, compute the distance from the threat center C_k to the projection of path segment $\overrightarrow{W_{ij}W_{i,j+1}}$, denoted as d_k.

Step 2) Compare the size of d_k and radius of the threat circle R_k. If $d_k \geq R_k$, the threat cost of the kth threat circle to the path segment $\overrightarrow{W_{ij}W_{i,j+1}}$ is zero, i.e., $J_{2,k}(\overrightarrow{W_{ij}W_{i,j+1}}) = 0$; else go to Step 3).

Step 3) Calculate the length of the projection $\overrightarrow{W'_{ij}W'_{i,j+1}}$ included in the kth threat circle, denoted as l_k. If $d_k \leq 1$, then $J_{2,k}(\overrightarrow{W_{ij}W_{i,j+1}}) = R_k l_k$; else $J_{2,k}(\overrightarrow{W_{ij}W_{i,j+1}}) = R_k l_k / d_k$.

Step 4) Threat cost of X_i can be expressed by

$$J_2(X_i) = \sum_{k=1}^{K} \sum_{j=0}^{n} s_k J_{2,k}\left(\overrightarrow{W_{ij}W_{i,j+1}}\right) \tag{25}$$

where K is the number of threats, and s_k indicates the threat intensity of the kth threat.

In view of the physical limitation of UAV, it can only turn and climb/dive with an angle less than or equal to a predetermined maximum turning and climbing/diving angle. Therefore, the turn and climb/dive angle constraint should be enforced at each path node where UAV is unlikely to make a sharp turn and climb/dive at such a point. Take two adjacent path segments $\overrightarrow{W_{ij}W_{i,j+1}}$ and $\overrightarrow{W_{i,j+1}W_{i,j+2}}$ as an example to define the turning angle. The turning angle at point $W_{i,j+1}$ is defined as the complementary angle of $\angle W'_{ij}W'_{i,j+1}W'_{i,j+2}$, i.e., $\angle\alpha_{ij}$ as shown in Fig. 5, where W'_{ij}, $W'_{i,j+1}$, and $W'_{i,j+2}$ are the corresponding projections of the path nodes W_{ij}, $W_{i,j+1}$, and $W_{i,j+2}$ in the xoy plane, respectively. Namely, the turning angle is the difference between the flight direction of the current path segment and that of the next one. The turning angle can be calculated according to the cosine theorem

$$\alpha_{ij} = \arccos \frac{\overrightarrow{W'_{ij}W'_{i,j+1}} \cdot \overrightarrow{W'_{i,j+1}W'_{i,j+2}}}{\left\| \overrightarrow{W'_{ij}W'_{i,j+1}} \right\| \left\| \overrightarrow{W'_{i,j+1}W'_{i,j+2}} \right\|} \tag{26}$$

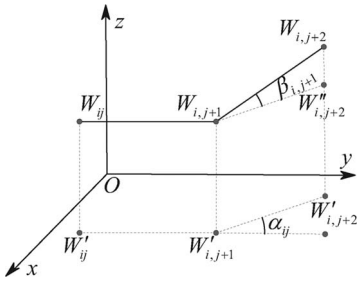

Fig. 5. Turning angle and climbing/diving angle.

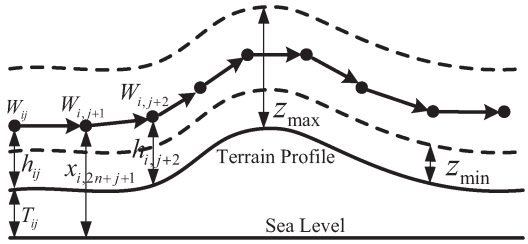

Fig. 6. Height cost computation.

where the symbol $\cdot$ indicates the inner product operation. This process yields in (27), shown at the bottom of the page.

Substituting (27) into (26), the turn angle α_{ij} can be expressed by the coordinates of W'_{ij}, $W'_{i,j+1}$, and $W'_{i,j+2}$.

The climb/dive angle of the path segment $\overrightarrow{W_{i,j+1}W_{i,j+2}}$ is defined as the angle between path segment itself and its projection in the horizontal plane, i.e., $\angle\beta_{i,j+1}$ as depicted in Fig. 5. In the right triangle $W_{i,j+1}W''_{i,j+2}W_{i,j+2}$ formed by path segment $\overrightarrow{W_{i,j+1}W_{i,j+2}}$ and its projection, the interior angle $\beta_{i,j+1}$ is given by

$$\beta_{i,j+1}=\arctan\left[(x_{i,2n+j+2}-x_{i,2n+j+1})\Big/\left\|\overrightarrow{W_{i,j+1}W''_{i,j+2}}\right\|\right]. \tag{28}$$

Clearly, the UAV climbs upwards when the altitude of path node $W_{i,j+2}$ is higher than that of $W_{i,j+1}$, which is equivalent to $x_{i,2n+j+2} > x_{i,2n+j+1}$ or $\beta_{i,j+1} > 0$, otherwise it dives downwards.

The turn angle cost and climb/dive angle cost are measured by the differences between the current turn and climb/dive angle and predetermined maximum turning and climbing/diving angle $\alpha_{\max}$ and $\beta_{\max}$. When the current turn angle is less than or equal to $\alpha_{\max}$, the cost of this turn angle is set to zero. Similarly, when the absolute value of the current climb/dive angle is less than or equal to $\beta_{\max}$, the cost of this climb/dive angle is also set to zero. In this way, the turn angle cost J_3 and climb/dive angle cost J_4 can be calculated according to

$$\begin{cases} J_3(X_i) = p_3 \sum_{j=0}^{n-1} d\alpha_{ij} \\ d\alpha_{ij} = \begin{cases} \alpha_{ij} - \alpha_{\max},\ \alpha_{ij} > \alpha_{\max} \\ 0,\ \alpha_{ij} \le \alpha_{\max} \end{cases} \end{cases} \tag{29}$$

$$\begin{cases} J_4(X_i) = p_4 \sum_{j=0}^{n} d\beta_{ij} \\ d\beta_{ij} = \begin{cases} |\beta_{ij}| - \beta_{\max},\ |\beta_{ij}| > \beta_{\max} \\ 0,\ |\beta_{ij}| \le \beta_{\max} \end{cases} \end{cases} \tag{30}$$

where α_{i0} and $\alpha_{i,n-1}$ are the turn angles formed by the waypoints $\{W_{i0}, W_{i1}, W_{i2}\}$ and $\{W_{i,n-1}, W_{i,n}, W_{i,n+1}\}$, respectively. Similarly, β_{i0} and $\beta_{i,n}$ are the climb/dive angles formed by the waypoints $\{W_{i0}, W_{i1}\}$ and $\{W_{i,n}, W_{i,n+1}\}$, respectively. Penalty factors p_3 and p_4 are two positive constants.

As discussed in Section V-B, while executing terrain-following flight, the UAV is required to follow the terrain with a minimum altitude clearance $z_{\min}$ to prevent the UAV from crashing. Furthermore, to reduce the probability of being detected by radar, the UAV should fly with a maximum altitude clearance $z_{\max}$. Denote the terrain elevation at the waypoint W_{ij} as T_{ij}, then the relative altitude of UAV at this point h_{ij} is the difference between the absolute altitude $x_{i,2n+j}$ and terrain elevation T_{ij}, i.e., $h_{ij} = x_{i,2n+j} - T_{ij}$. As illustrated in Fig. 6, the height cost J_5 can be calculated according to

$$\begin{cases} J_5(X_i) = p_5 \sum_{j=0}^{n+1} dh_{ij} \\ dh_{ij} = \begin{cases} h_{ij} - z_{\max}, h_{ij} > z_{\max} \\ 0, z_{\min} \le h_{ij} \le z_{\max} \\ z_{\min} - h_{ij}, 0 < h_{ij} < z_{\min} \\ \infty, h_{ij} \le 0 \end{cases} \end{cases} \tag{31}$$

where p_5 is a positive penalty factor and the symbol ∞ represents infinity. Note that $h_{ij} \le 0$ means the UAV has hit the ground.

D. Route Planning Using θ-QPSO

The detailed implementation of route planning based on the θ-QPSO algorithm can be described as follows:

Step 1) Choose appropriate parameters such as population size, particle dimension, and maximum number of iterations.

Step 2) Input the environmental information of war field and selected parameters in Step 1).

Step 3) Initialize the phase angle of each particle randomly in the interval of $[-\pi/2, \pi/2]$ to generate initial coarse paths.

Step 4) Using the mapping f defined in (22) to calculate the position corresponding to phase angle.

Step 5) Evaluate each path based on the aforementioned cost function defined in (23).

Step 6) Compute Λ_i and Λ_g using (10) and (11), where Λ_i and Λ_g are the phase angle of the ith particle's personal best $(pbest)$ position and the phase angle of the swarm's global best $(gbest)$ position, respectively.

Step 7) Compute Γ_i and Ψ according to (9), where Γ_i and Ψ are the phase angle of the each particle's local attractor and the phase angle of the swarm's mean best $(mbest)$ position, respectively.

Step 8) Renew the phase angle of each particle in the swarm according to (9) and confine them to $[-\pi/2, \pi/2]$.

Step 9) Repeat Steps 4) to 8) until the maximum number of iterations is reached.

Step 10) Output $f(\Lambda_g)$ as the optimal path when the loop ends.

$$\begin{cases} \overrightarrow{W'_{ij}W'_{i,j+1}} \cdot \overrightarrow{W'_{i,j+1}W'_{i,j+2}} = (x_{i,j+1} - x_{ij})(x_{i,j+2} - x_{i,j+1}) \\ +(x_{i,n+j+1} - x_{i,n+j})(x_{i,n+j+2} - x_{i,n+j+1}) \\ \left\|\overrightarrow{W'_{ij}W'_{i,j+1}}\right\| = \sqrt{(x_{i,j+1} - x_{ij})^2 + (x_{i,n+j+1} - x_{i,n+j})^2} \\ \left\|\overrightarrow{W'_{i,j+1}W'_{i,j+2}}\right\| = \sqrt{(x_{i,j+2} - x_{i,j+1})^2 + (x_{i,n+j+2} - x_{i,n+j+1})^2}. \end{cases} \tag{27}$$

VI. Experimental Results

The route planning for UAV based on the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO algorithms was implemented in a Matlab 7.6 programming environment on a PC with Intel Core2 Duo E7200 CPU running Windows XP. Suppose the war field to be a square region with a size of 512 pixel $\times$ 512 pixel and with a resolution of 100 m $\times$ 100 m for each pixel.

Unless otherwise specified, the following parameter values are used in our route planning experiments. The population size, number of path nodes, and maximum number of iterations are set to 20, 5, and 100, respectively. All route planning experimental results are the average values of 30 runs. The weight coefficients in the cost function determine the relative emphasis of various components with respect to the overall cost function and are all set to be 0.2 so as to ensure all weights sum up to 1.0 in total and produce a uniform effect of the five terms in the cost value. The minimum and maximum safe clearance between UAV and terrain are set to the values $z_{\min} = 20$ m and $z_{\max} = 100$ m. The maximum turning angle and climbing/diving angle are set to $\alpha_{\max} = 30^\circ$ and $\beta_{\max} = 30^\circ$. The settings of $z_{\min}$, $z_{\max}$, $\alpha_{\max}$, and $\beta_{\max}$ are dependent on the physical limitation of UAV and can be considered as the mission parameters rather than algorithm ones [1]–[3].

For convenience, we distinguish between the public algorithm parameters and private algorithm parameters. The public algorithm parameters are defined as the common parameters of the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO. There are the following four public algorithm parameters: population size, particle dimension (equivalent to the number of path nodes), maximum number of iterations, and number of runs. As mentioned in Section V-B, the PSO-based algorithms are not sensitive to the population size, and particle dimension is a tradeoff between accuracy of the path and computational efficiency. Consequently, the population size remains unchanged, and the number of path nodes is set to be 5, which means the particle dimension is 15.

When solving real-world problems, usually the fitness calculation accounts for the most time since the PSO-based algorithms are highly computation efficient. The proposed θ-QPSO does not impose any additional burden in terms of objective function evaluations, since it has the same number of objective function evaluations as GA, DE, and three other PSO-based algorithms. For the path planning problem, the cost function calculation is more complex than the benchmark function fitness computation. Therefore, the maximum number of iterations and number of trials are both reduced relatively compared to those values used in Section IV.

Meanwhile, each algorithm has its own particular parameters, which are defined as private algorithm parameters, including the crossover and mutation probability of GA, mutation factor and crossover constant of DE, inertial weight coefficient of PSO and θ-PSO, and contraction-expansion coefficient of QPSO and θ-QPSO. The settings of these private algorithm parameters are the same as those used in Section IV, which are selected based on the suggestions in other literature [13], [14], [16], [27], [49], [50] where these values have been found, empirically, to provide good performance.

Two test cases are designed to compare the performance of the GA, DE, and four PSO-based algorithms in the presence of different battlefield environments, which mainly mean different start points, endpoints, and threat environments. The information of these two test cases is described in Table IV. The first column of Table IV represents the test case number. As noted above (Section V-C), the threats are modeled by circles with different centers and radii. Threat (99,302,47) means the center of this threat is (99,302) with a radius of 47. The start point and endpoint are depicted by their spatial coordinates.

TABLE IV
Description of Test Cases

Case	Start point	Endpoint	Threats
1	(13,502,149)	(503,108,200)	{(99,302,47),(246,140,60), (287,246,59),(325,357,67)}
2	(8,10,146)	(507,505,191)	{(182,139,53),(266,285,68), (376,245,60),(117,50,38),(219,387,68)}

For the purpose of comparison, the best paths' projections in the xoy plane of case 1 generated by the GA (dashed thin line), DE (solid thin line), PSO (dashed thick line), θ-PSO (dash-dot thick line), QPSO (dotted thick line), and θ-QPSO (solid thick line) during 30 runs are showed in Fig. 7(a). The solid pentagram and diamond are the start point and endpoint of the path, respectively. Synthetic threats are modeled as concentric circles. The "•" represents the path node on the generated path. Note that the GA and PSO failed to find a safe path and the resultant paths cut through the threat under the same parameter setting conditions, which means the GA and PSO are both prone to be trapped in local optima. The route planners based on the DE, θ-PSO, QPSO, and θ-QPSO all can successfully generate threat-avoidance paths. Moreover, the turn angle cost, climb/dive angle cost, and height cost of the best paths generated by the DE, θ-PSO, QPSO, and θ-QPSO during 30 runs are all equal to 0. According to (29)–(31) and the definition of constraints in Section I, it is concluded that the best paths generated by the DE, θ-PSO, QPSO, and θ-QPSO satisfy all constraints.

The relationship between number of iterations and average best fitness values over 30 runs is showed in Fig. 7(b). Note that the average best fitness value is defined as the reciprocal of the mean best cost value. It is easy to be seen from Fig. 7(b) that θ-QPSO has not only stronger global searching ability, but also faster convergence speed than the GA, DE, and three other PSO-based algorithms according to the discussion of search ability and convergence speed given in Section IV.

In fact, as mentioned in Section IV, the convergence curve visually displays the convergence speed of an algorithm. Moreover, for two given algorithms, we can judge which algorithm converges faster than the other one through the location relationship between the convergence curves of them. Because the average best fitness is defined as the inverse of the mean best cost, the convergence curves illustrated in Fig. 7(b) are ascending while curves shown in Figs. 2 and 3 are descending. Consequently, for UAV path planning problem, if the convergence curve of algorithm A is always above that of algorithm B, then the convergence rate of algorithm A is strictly faster than that of algorithm B, which is just the opposite of the situation that occurred in Section IV. As shown in Fig. 7(b), the convergence curve of θ-QPSO is at the very top of Fig. 7(b), which means the θ-QPSO converges fastest among the six algorithms. It is worthwhile to note that the same is true for Figs. 9(b) and 11(b).

During 30 times experiments, the minimum, maximum, average, and standard deviation of cost value are recorded in Table V. The route planner based on θ-QPSO possesses the minimum mean cost and standard deviation, which further demonstrates the high performance and good robustness of the proposed θ-QPSO.

The 3-D stereo displays and vertical profiles corresponding to the best paths in Fig. 7(a) are shown in Fig. 8, where (a)–(f) are the 3-D paths generated by the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO, respectively; (g)–(l) are the corresponding vertical profiles of (a)–(f). Following the suggestion of Zheng *et al.* [3], the coordinate axes of each plot in Fig. 8 are omitted. The same is true for Figs. 10 and 12.

In the path vertical profiles figures, the upper curves and lower shaded regions are the path profiles and terrain contours, respectively.

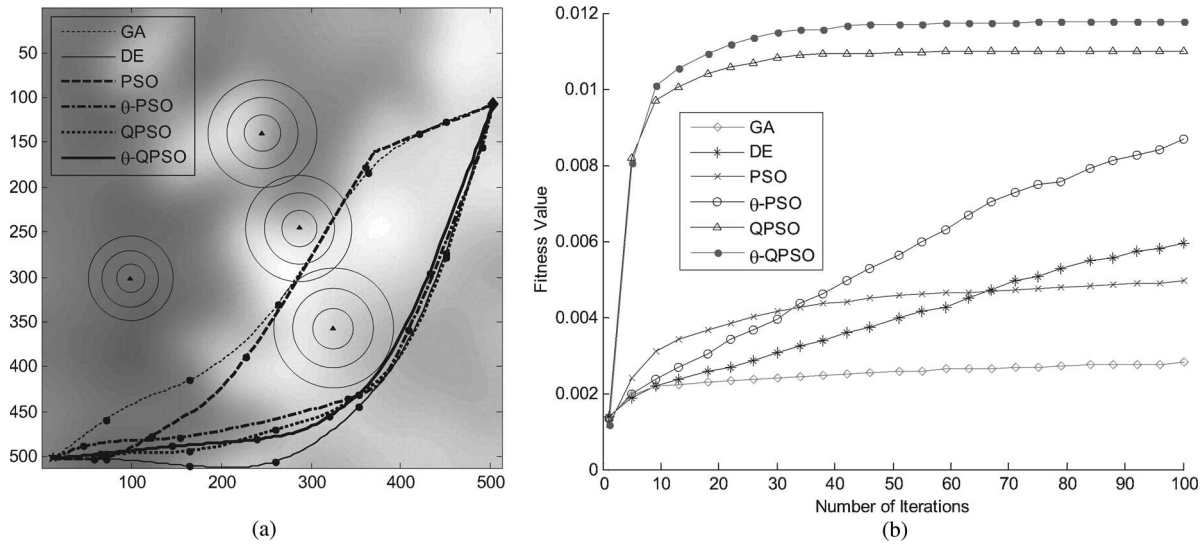

Fig. 7. Comparison of results produced by the GA, DE, and four PSO-based algorithms on case 1: (a) path projections in the xoy plane; (b) convergence curves of average best fitness values.

TABLE V
PERFORMANCE COMPARISON OF SIX ALGORITHMS ON CASE 1

Algorithm	Min cost	Max cost	Mean cost	St Dev
GA	161.2103	575.9078	352.6507	106.9226
DE	92.9672	308.1928	167.9151	69.7142
PSO	150.9860	294.2757	201.1002	34.8624
θ-PSO	77.7264	313.2053	115.1755	44.5729
QPSO	76.5698	118.3859	90.8255	10.0096
θ-QPSO	74.7547	95.9719	84.9559	4.4422

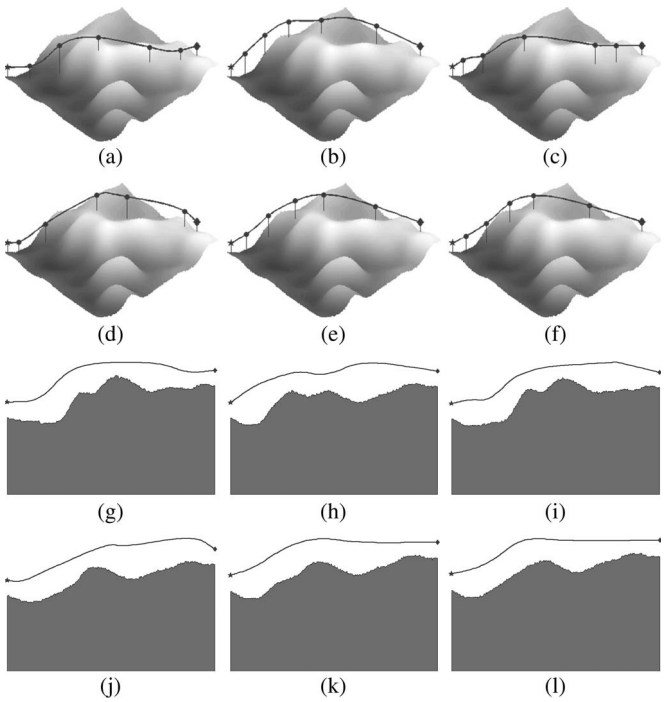

Fig. 8. Three-dimensional stereo displays and vertical profiles corresponding to the best paths in Fig. 7(a): (a)–(f) are the 3-D paths generated by the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO; (g)–(l) are the corresponding vertical profiles of (a)–(f).

As can be seen from Fig. 8, the best paths produced by QPSO and θ-QPSO can follow the terrain much better than those generated by the GA, DE, PSO, and θ-PSO. There is no doubt that threat-avoidance and terrain-following path increases both the survivability of the UAV and the effectiveness of mission.

Figs. 9 and 10 display the experimental results of case 2, and corresponding experimental data is recorded in Table VI. These experimental results demonstrate that the θ-QPSO still maintains its superiority compared with the GA, DE, and three other PSO-based algorithms.

From the statistical results shown in Table VI, we observe that the mean cost value and the corresponding standard deviation of the θ-QPSO are both less than those of the other five algorithms. As mentioned in Section IV, the search ability and stability of an algorithm can be reflected by the mean fitness value and the standard deviation value, respectively. Therefore, it is concluded that the θ-QPSO has better search ability and robustness than the GA, DE, and three other PSO-based algorithms. Meanwhile, it is not hard to find that the θ-QPSO achieves the fastest convergence speed among the six algorithms according to the position relationship of convergence curves shown in Fig. 9(b).

Under the same parameter setting conditions, the GA and PSO failed to find a safe path and their resultant paths passed through the threat. It is easy to find that the solid thick line generated by θ-QPSO is shorter and smoother than those generated by the DE (solid thin line), θ-PSO (dash-dot thick line), and QPSO (dotted thick line). In other words, the path produced by θ-QPSO is with less cost and more flyable. Moreover, the route planners based on QPSO and θ-QPSO guide the UAV at a low altitude, and the minimum clearance between UAV and ground is satisfied, whereas the route planners based on PSO and θ-PSO weave their paths through threat with bad terrain-following effect.

Similar experiment results obtained on case 1 and case 2 testify that the performance of PSO-based algorithms is independent of changed battlefield environments, which mainly refer to different start points, endpoints, and threat environments.

As mentioned before (Section V-B), the number of path nodes n (equivalent to the dimension of a particle) has a very close relationship with accuracy of the path and computational efficiency. Greater number of path nodes results in more computational time and slower

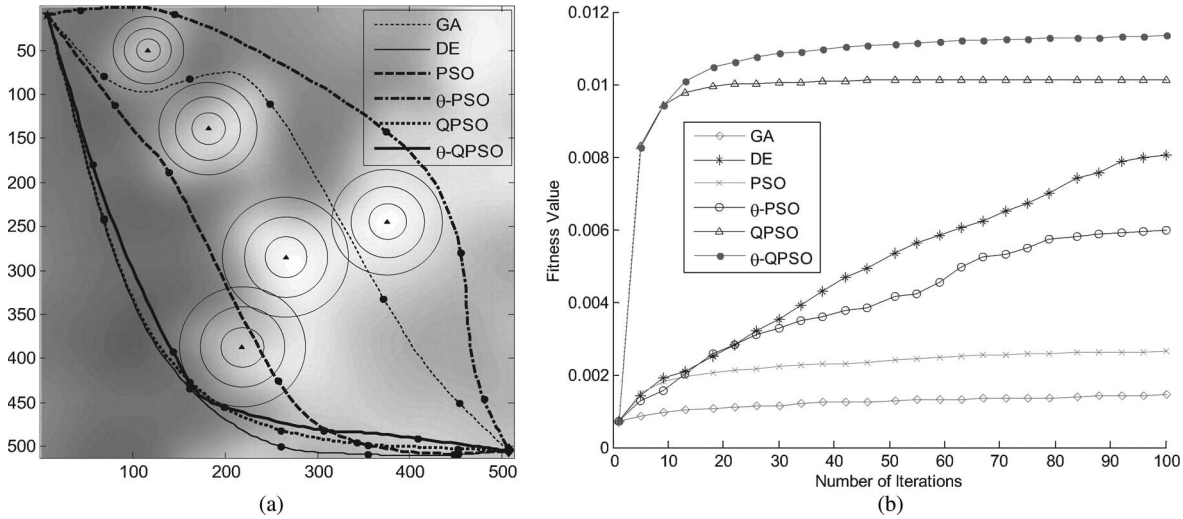

Fig. 9. Comparison of results produced by the GA, DE, and four PSO-based algorithms on case 2: (a) path projections in the xoy plane; (b) convergence curves of average best fitness values.

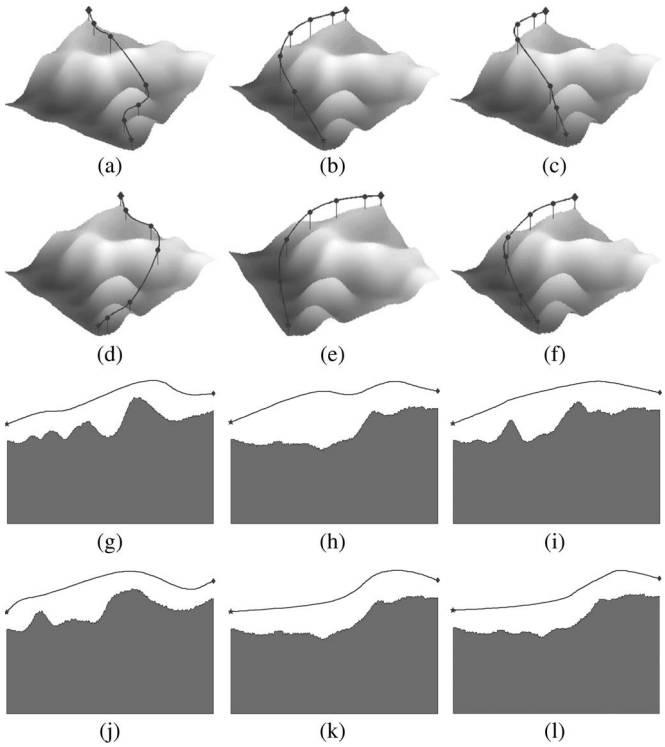

Fig. 10. Three-dimensional stereo displays and vertical profiles corresponding to the best paths in Fig. 9(a): (a)–(f) are the 3-D paths generated by the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO; (g)–(l) are the corresponding vertical profiles of (a)–(f).

convergence speed. Theoretically, other conditions being equal, the larger the dimension of the particle, the more accurate the path will be, and the more complicate of the problem becomes. Therefore, too large value of dimension may result in failure of planning. Figs. 11 and 12 show the experimental results of case 2, while the number of path nodes is set to be 8 and the corresponding experimental data is recorded in Table VII.

Note that the dimension of a particle is three times as large as the number of path nodes. Therefore, the dimension of each particle increases from 15 to 24 when the number of path nodes increases from 5 to 8. Compared with the results given in Table VI, all four statistics of each algorithm shown in Table VII are increased relatively, which is

TABLE VI
PERFORMANCE COMPARISON OF SIX ALGORITHMSON CASE 2

Algorithm	Min cost	Max cost	Mean cost	St Dev
GA	421.9394	1167.8226	690.0622	162.0791
DE	93.2824	216.5375	123.9247	33.6912
PSO	181.2812	471.9321	379.6359	74.1472
θ-PSO	85.1249	551.6959	167.5276	125.6694
QPSO	84.9523	160.7084	98.6591	19.1128
θ-QPSO	78.7782	124.8747	88.0872	7.9850

mainly because the number of path nodes has been increased to 8. That is to say, as the dimension of the particle increases, the problem to be solved becomes more and more complex. Therefore, it is more difficult for the GA, DE, and all four PSO-based algorithms to find the global optimal solution of the problem at hand in high-dimensional search space compared with low-dimensional solution space. However, the θ-QPSO still has an advantage over the GA, DE, and three other PSO-based algorithms.

It should be noted that the performance of DE on case 2 with $n = 8$ is dramatically different from its performance on case 2. The DE achieves better results than PSO and θ-PSO on case 2, while the results obtained by DE on case 2 with $n = 8$ are inferior to all four PSO-based algorithms according to the mean cost values and standard deviation values recorded in Tables VI and VII. Such a significant change in the performance of DE indicates that DE is sensitive to the number of path nodes for the UAV route planning application.

It should also be noted that the performance of GA on case 1, case 2, and case 2 with $n = 8$ are all poor and far more perfect. As can be seen from Tables V–VII, the four statistics (i.e., the minimum cost, maximum cost, mean cost, and the standard deviation) of GA are all the biggest among the six algorithms. Moreover, as illustrated in Fig. 7(a), Fig. 9(a), and Fig. 11(a), the path planners based on GA for case 1, case 2, and case 2 with $n = 8$ all failed to produce a threat-avoidance path.

By comparing the convergence curves shown in Figs. 9(b) and 11(b), it is easy to see that as a result of the increase of particle dimension, the convergence speeds of the GA, DE, and all four PSO-based algorithms slow down. However, the θ-QPSO still outperforms the other three PSO-based algorithms after about 45 times iterations and eventually finds a better path with less path cost. More interesting, the route planner based on PSO finds a longer path with less exposure to threat in Fig. 11(a), while a shorter but with high threat cost path

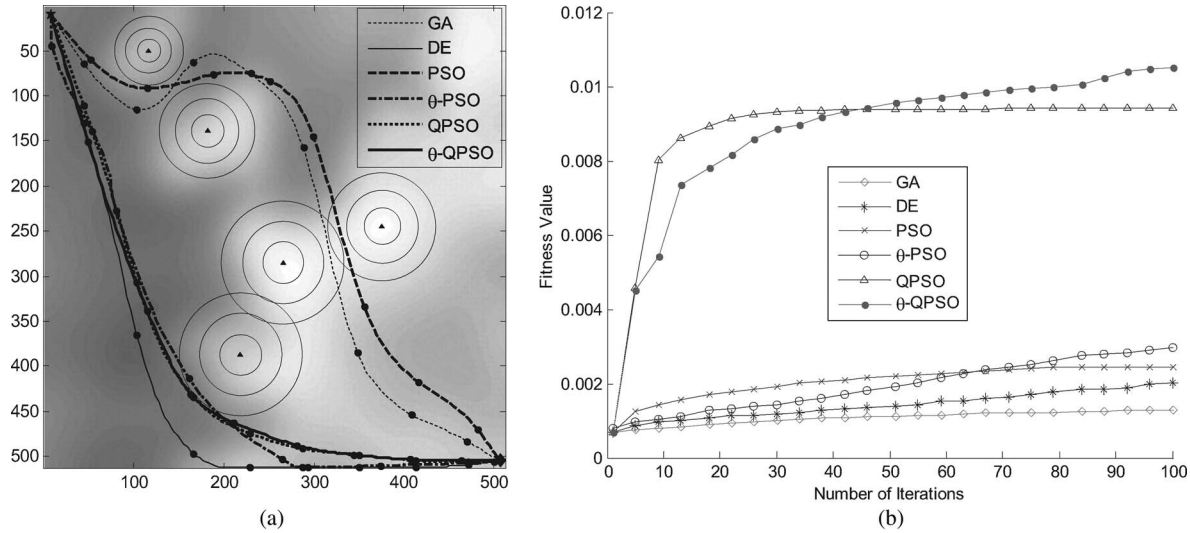

Fig. 11. Comparison of results produced by the GA, DE, and four PSO-based algorithms on case 2 with $n = 8$: (a) path projections in the xoy plane; (b) convergence curves of average best fitness values.

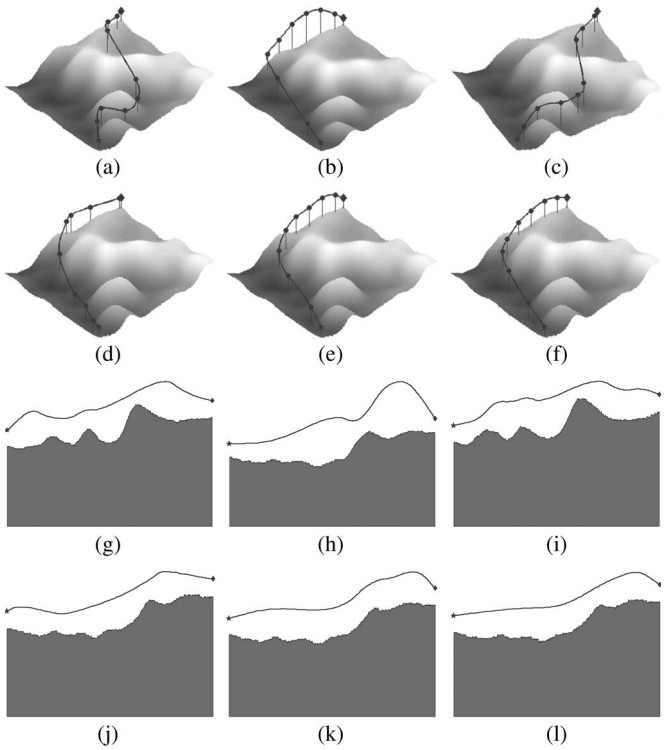

Fig. 12. Three-dimensional stereo displays and vertical profiles corresponding to the best paths in Fig. 11(a): (a)–(f) are the 3-D paths generated by the GA, DE, PSO, θ-PSO, QPSO, and θ-QPSO; (g)–(l) are the corresponding vertical profiles of (a)–(f).

is obtained in Fig. 9(a). Except for the paths generated by the GA and PSO, the other four paths shown in Fig. 11 all successfully avoid the threats. Meanwhile, as illustrated in Fig. 12, except for the path generated by DE, the paths produced by the GA and four PSO-based algorithms all successfully fulfilled the nap-of-the-earth guidance of UAV, although the terrain-following effect of the GA and three other PSO-based algorithms is not as good as that of the θ-QPSO algorithm.

VII. Conclusion and Future Work

This paper proposed a new variant of PSO, which is called phase angle-encoded and quantum-behaved particle swarm optimization (θ-QPSO). Based on the results of six algorithms on different benchmark functions, it is concluded that θ-QPSO significantly improves the PSO's performance and gives the best performance on all six test functions no matter they are unimodal or multimodal when compared with the GA, DE, and three other PSO versions. At the same time, the effectiveness and performance of the GA, DE, and four PSO-based algorithms for the route planning application were compared based on the solution quality (the cost value or fitness value of generated path), stability, and convergence speed of the six algorithms. The proposed θ-QPSO algorithm has shown its high performance in solving path planning problem for UAV in different known and static threat environments. The experimental results demonstrated that the θ-QPSO algorithm can not only obtain better solution, but also improve the robustness and speed up the convergence of PSO, while the GA, DE, and three other PSO-based algorithms may lead to premature convergence and become trapped in local optima during the evolution. Compared with the path planners based on the GA, DE, and three other PSO-based algorithms, the path planner based on the θ-QPSO algorithm can produce more optimal path at a faster convergence speed. Most crucially, the 3-D paths generated by the θ-QPSO algorithm under different complex battlefield circumstances can realize threat avoidance and terrain-following control effectively, while at the same time satisfying UAV's maneuvering performance constraints.

As for future research, the current work can be extended in the following directions. First, the convergence property and convergence speed of the proposed θ-QPSO should be systematically investigated with more rigorous theoretical analysis. Another extension of this work could be to compare the performance of the θ-QPSO with other classical path planning approaches mentioned in Section I, not just confined to the performance comparison among evolutionary algorithm family.

TABLE VII
Performance Comparison of Six Algorithms on Case 2 With $n = 8$

Algorithm	Min cost	Max cost	Mean cost	St Dev
GA	443.5172	1394.1312	780.4026	218.8872
DE	205.1221	889.9771	493.8382	172.6894
PSO	236.1218	722.3512	407.1976	115.9822
θ-PSO	103.4087	637.2100	338.7810	128.2931
QPSO	87.8542	156.8518	106.3278	19.0634
θ-QPSO	81.1170	255.1075	95.1394	31.4155

Finally, the cooperating path planning problem of multiple UAVs can be considered in the future.

ACKNOWLEDGMENT

The authors acknowledge the help of C. Zheng in the preparation of this paper. The authors would also like to thank the anonymous reviewers for their valuable comments and constructive suggestions, which helped improve the quality of this paper.

REFERENCES

[1] C. Zheng, M. Ding, and C. Zhou, "Real-time route planning for unmanned air vehicle with an evolutionary algorithm," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 17, no. 1, pp. 63–81, Feb. 2003.

[2] C. Zheng, M. Ding, C. Zhou, and L. Li, "Coevolving and cooperating path planner for multiple unmanned air vehicles," *Eng. Appl. Artif. Intell.*, vol. 17, no. 8, pp. 887–896, Dec. 2004.

[3] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 609–620, Aug. 2005.

[4] R. V. Helgason, J. L. Kennington, and K. R. Lewis, "Cruise missile mission planning: A heuristic algorithm for automatic path generation," *J. Heuristics*, vol. 7, no. 5, pp. 473–494, Sep. 2001.

[5] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Proc. 4th Int. Symp. Voronoi Diagrams Sci. Eng.*, 2007, pp. 38–47.

[6] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Trans. Robot. Autom.*, vol. 18, no. 6, pp. 911–922, Dec. 2002.

[7] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative-timing missions," *J. Guid. Control Dyn.*, vol. 28, no. 1, pp. 150–161, Jan. 2005.

[8] D. Eppstein, "Finding the k shortest paths," *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, Apr. 1999.

[9] P. O. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned helicopter," *J. Intell. Fuzzy Syst.*, vol. 17, no. 4, pp. 395–405, Sep. 2006.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[11] R. Szczerba, P. Galkowski, I. Glicktein, and N. Ternullo, "Robust algorithm for real-time route planning," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 3, pp. 869–878, Jul. 2000.

[12] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1994, pp. 3310–3317.

[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[14] M. Yi, M. Ding, and C. Zhou, "3D route planning using genetic algorithm," in *Proc. SPIE Int. Symp. Multispectral Image Process.*, 1998, pp. 92–95.

[15] I. Nikolos, K. Valavanis, N. Tsourveloudis, and A. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 6, pp. 898–912, Dec. 2003.

[16] I. Nikolos, E. Zografos, and A. Brintaki, "UAV path planning using evolutionary algorithms," in *Innovations in Intelligent Machines*, S. C. Javaan, C. J. Lakhmi, M. Akiko, and S.-I. Mika, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 77–111.

[17] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[18] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[19] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[20] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[21] P. Larranaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Dordrecht, The Netherlands: Kluwer, 2002.

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

[23] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Evolutionary Programming VII*, V.W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 601–610.

[24] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. Congr. Evol. Comput.*, 2001, pp. 81–86.

[25] G. G. Yen and L. Wen Fung, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 4, pp. 890–911, Jul. 2009.

[26] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 69–73.

[27] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1945–1950.

[28] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proc. Congr. Evol. Comput.*, 2001, pp. 101–106.

[29] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proc. Congr. Evol. Comput.*, 2001, pp. 94–100.

[30] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 591–600.

[31] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317–325, Mar. 2003.

[32] Y.-L. Zheng, L.-H. Ma, L.-Y. Zhang, and J.-X. Qian, "On the convergence analysis and parameter selection in particle swarm optimization," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2003, pp. 1802–1807.

[33] M. Jiang, Y. P. Luo, and S. Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Inf. Process. Lett.*, vol. 102, no. 1, pp. 8–16, Apr. 2007.

[34] C. Lin and Q. Feng, "The standard particle swarm optimization algorithm convergence analysis and parameter selection," in *Proc. Int. Conf. Nat. Comput.*, 2007, pp. 823–826.

[35] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[36] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1931–1938.

[37] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1958–1962.

[38] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 4, pp. 515–519, Jul. 2006.

[39] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 1997, pp. 4104–4108.

[40] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 84–89.

[41] N. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Proc. IEEE Swarm Intell. Symp.*, 2003, pp. 72–79.

[42] A. Stacey, M. Jancic, and I. Grundy, "Particle swarm optimization with mutation," in *Proc. Congr. Evol. Comput.*, 2003, pp. 1425–1430.

[43] M. Lovbjerg, T. Rasmussen, and T. Krink, "Hybrid particle swarm optimiser with breeding and subpopulations," in *Proc. 3rd GECCO*, 2001, pp. 469–476.

[44] B. Liu, L. Wang, Y. Jin, F. Tang, and D. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons Fractals*, vol. 25, no. 5, pp. 1261–1271, Jul. 2005.

[45] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[46] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, and S.-J. Ho, "OPSO: Orthogonal particle swarm optimization and its application to task assignment problems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 2, pp. 288–298, Mar. 2008.

[47] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 211–224, Jun. 2004.

[48] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.

[49] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proc. Congr. Evol. Comput.*, 2004, pp. 325–331.

[50] J. Sun, W. Xu, and B. Feng, "A global search strategy of quantum-behaved particle swarm optimization," in *Proc. IEEE Conf. Cybern. Intell. Syst.*, 2004, pp. 111–116.

[51] S. M. Mikki and A. A. Kishk, "Quantum particle swarm optimization for electromagnetics," *IEEE Trans. Antennas Propag.*, vol. 54, no. 10, pp. 2764–2775, Oct. 2006.

[52] Y. Fu, M. Ding, C. Zhou, C. Cai, and Y. Sun, "Path planning for UAV based on quantum-behaved particle swarm optimization," in *Proc. SPIE Int. Symp. Multispectral Image Process. Pattern Recognit.*, 2009, vol. 7497B, pp. 1–7.

[53] L. d. S. Coelho and V. C. Mariani, "Particle swarm approach based on quantum mechanics and harmonic oscillator potential well for economic load dispatch with valve-point effects," *Energy Convers. Manage.*, vol. 49, no. 11, pp. 3080–3085, Nov. 2008.

[54] J. Sun, W. Fang, D. Wang, and W. Xu, "Solving the economic dispatch problem with a modified quantum-behaved particle swarm optimization method," *Energy Convers. Manage.*, vol. 50, no. 12, pp. 2967–2975, Dec. 2009.

[55] S. L. Sabat, L. d. S. Coelho, and A. Abraham, "MESFET DC model parameter extraction using quantum particle swarm optimization," *Microelectron. Reliab.*, vol. 49, no. 6, pp. 660–666, Jun. 2009.

[56] S. N. Omkar, R. Khandelwal, T. V. S. Ananth, G. Narayana Naik, and S. Gopalakrishnan, "Quantum behaved particle swarm optimization (QPSO) for multi-objective design optimization of composite structures," *Expert Syst. Appl.*, vol. 36, no. 8, pp. 11 312–11 322, Oct. 2009.

[57] J. Sun, X. Wu, W. Fang, Y. Ding, H. Long, and W. Xu, "Multiple sequence alignment using the Hidden Markov Model trained by an improved quantum-behaved particle swarm optimization," Inf. Sci., Nov. 2010.

[58] J. Sun, W. Fang, X. Wu, Z. Xie, and W. Xu, "QoS multicast routing using a quantum-behaved particle swarm optimization algorithm," *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 123–131, Feb. 2011.

[59] L. d. S. Coelho, "A quantum particle swarm optimizer with chaotic mutation operator," *Chaos, Solitons Fractals*, vol. 37, no. 5, pp. 1409–1418, Sep. 2008.

[60] L. d. S. Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1676–1683, Mar. 2010.

[61] W. Zhong, S. Li, and F. Qian, "θ-PSO: A new strategy of particle swarm optimization," *J. Zhejiang Univ. (Sci. A)*, vol. 9, no. 6, pp. 786–790, Jun. 2008.

[62] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. Congr. Evol. Comput.*, 2000, pp. 84–88.