Information Sciences xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Cellular particle swarm optimization

Yang Shi [a], Hongcheng Liu [a], Liang Gao [a,*], Guohui Zhang [b]

[a] The State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, PR China
[b] School of Management Science and Engineering, Zhengzhou Institute of Aeronautical Industry Management, Zhengzhou 450015, PR China

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Cellular particle swarm optimization
Cellular automata
Particle swarm optimization
Function optimization

ABSTRACT

This paper proposes a cellular particle swarm optimization (CPSO), hybridizing cellular automata (CA) and particle swarm optimization (PSO) for function optimization. In the proposed CPSO, a mechanism of CA is integrated in the velocity update to modify the trajectories of particles to avoid being trapped in the local optimum. With two different ways of integration of CA and PSO, two versions of CPSO, i.e. CPSO-inner and CPSO-outer, have been discussed. For the former, we devised three typical lattice structures of CA used as neighborhood, enabling particles to interact inside the swarm; and for the latter, a novel CA strategy based on "smart-cell" is designed, and particles cast their eyes to the outside of the swarm. Theoretical studies are made to analyze the convergence of CPSO, and numerical experiments are conducted to compare the proposed algorithm with different variants of PSO. According to the experimental results, the proposed method performs better than other variants of PSO on benchmark test functions.

1. Introduction

The development of our society presents us more and more complex real-world problems in scientific research, manufacturing, economic, etc. How to solve these problems at lower time costs and at less economic costs plays a crucial role in pushing forward the development of science and engineering, thus optimization has become an active but challenging field. In the sphere of optimization, function optimization is a widely discussed research topic, because many real-world problems in optimization could be abstracted and modeled as function optimization problems, and they serve as benchmark problems for tests on optimization algorithms. For solving function optimization problems, we propose to hybridize particle swarm optimization (PSO) with cellular automata (CA).

PSO, originally introduced by Kennedy and Eberhart [12], has become one of the most important swarm intelligence-based algorithms. It attracts increasing attention as a new optimization technique for solving complex optimization problems [17]. The unique information diffusion and interaction mechanism of PSO enable it to solve many problems with good performance at low computational cost. However, like all other swarm intelligence-based algorithms, escaping from the local optimum and preventing premature convergence are two inevitable difficulties when implementing PSO, especially when dimensionality increases, problems become more complex, and the possibility for finding global optimum sharply decreases.

CA is a discrete dynamical system that can produce complex phenomena [29]. The essential idea of the CA model is to simulate macro-behavior by the micro-dynamics, which is produced by the interaction of a population of individuals (cells) that are connected in particular neighborhood structures.

The paper employs a CA mechanism to improve the performance of PSO by devising two versions of cellular particle swarm optimization (CPSO): CPSO-inner and CPSO-outer. The CPSO-inner uses the information inside the particle swam

* Corresponding author. Tel.: +86 27 87559419; fax: +86 27 87543074.
E-mail address: gaoliang@mail.hust.edu.cn (L. Gao).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.05.025

to interact by considering every particle as a cell; whereas, the CPSO-outer enables cells that belong to the particle swarm to communicate with cells outside the particle swarm, every potential solution is defined as a cell and every particle of the swarm is defined as a smart-cell.

The structure of the paper is organized as follows: a brief overview of PSO is presented in Section 2. In Section 3, two versions of CPSO (CPSO-inner and CPSO-outer) are presented. Subsequently, theoretical studies are conducted to analyze the search behavior of CPSO, and a case study is presented to show the trajectory of particles in CPSO. Section 4 analyzes the computational complexity of several variants of PSO and compiles a large set of benchmark test functions to systematically examine the performance of CPSO through computational results of these functions. Finally, conclusions and further discussions are given in Section 5.

2. Particle swarm optimization

Due to its simple implementation, minimum mathematical processing and good optimization capability, PSO has attracted more attentions. As Vassiliadis and Dounias [37] summarized, PSO can be considered the most popularly and frequently applied among nature-inspired techniques according to related publications.

PSO simulates the behavior of a bird flock. When a flock of birds forage for food, two simple and important strategies are: (a) searching for the peripheral region around the bird that is nearest to the food; (b) judging the position of the food by its own flying experience. Inspired by the two strategies, the search space of optimization problem is regarded as birds' flying space; every bird is abstracted as a particle with non-quality and non-volume so as to denote a candidate solution; and the optimal solution to be searched for the problem is the food to all the particles. Then, the basic PSO algorithm comprises a swarm of particles moving in the D-dimensional search space which includes all possible candidate solutions.

We denote the ith particle as $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, and its flying velocity as $V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$. Depending on two strategies above, when each particle "flies" in the search space, $P_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,d})$ denotes the personal best position the ith particle has found so far, and $P_g = (p_{g,1}, p_{g,2}, \ldots, p_{g,d})$ is the global best position discovered by the swarm. At each time step t, both of each particle's velocity and position are updated so that a particle moves to a new position. The following two equations are employed to calculate the velocity and position:

$$V_i^{t+1} = V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t) \quad (1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2)$$

where c_1 and c_2 are two positive constants (acceleration constants), r_1 and r_2 are two uniform random numbers in $[0,1]$. A constant $V_{\max}$ is often used to limit each particle's velocity to guarantee that most of the new positions of the particles will be restricted in the search space of feasible region at each iteration. Eqs. (1) and (2) lead to the movement of each particle towards its cognitive best position (P_i) and "social" best position (P_g) with random perturbations caused by $c_1 r_1$ and $c_2 r_2$, respectively. The quality of a particle's position is measured by its fitness value, namely, the better fitness value means the better position. In this paper, we discuss minimization problems, that is, the smaller the objective value, the better the particle's position and fitness value. Of course, when solving maximization problems, we could transform the maximization problem into minimization problem.

Shi and Eberhart [31] modified the original PSO by introducing an inertia weight (ω) to Eq. (1) to balance exploitation and exploration. The modified Eq. (1) is as follows:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 \left(P_g^t - X_i^t\right) \quad (3)$$

According to the numerical tests in [31,32], the adoption of ω improves the performance of PSO largely, so the combination of Eqs. (3) and (2) is always considered as the standard PSO by many researchers. Fig. 1 is a schematic drawing that illustrates how the position of a particle updates, and Fig. 2 presents the pseudo-code of standard PSO.

Since PSO was proposed, investigations have been made theoretically and experimentally to analyze and improve PSO. Ozcan and Mohan [24] were the first to analyze PSO in theoretical aspects, and the trajectories of particles were analyzed in one dimension under a simplified model. Following this, the theoretical analysis was further conducted in multi-dimensional search space [25], and analytical results show that particles oscillate in a sinusoidal-wave manner. Clerc and Kennedy [4] explored how PSO works from a mathematical perspective, introduced a constriction factor χ to guarantee the convergence of PSO, and analyzed the trajectory of a single particle in both discrete time and continuous time. Van den Bergh and Engelbrecht [36] analyzed how the inertia weight and acceleration constants affect the trajectories of particles and provided theoretical findings on the dynamics of the PSO systems. These studies provided theoretical supports for the research on the improvement of PSO.

In order to enhance the performance of PSO, attempts have been made, including restricting the value of velocity, integrating mechanisms of different optimization algorithms, studying neighborhood topology of particles in the swarm of PSO, and developing comprehensive learning strategies.

Following the original idea on particle swarm attempting to discover a novel optimization mechanism through simulation of a social model, Kennedy and Eberhart [13] proposed a discrete binary PSO. This variant of PSO adopted Sigmoid function to restrict the value of each dimension of a velocity in $[0,1]$. The Sigmoid function was applied to denote the probability at

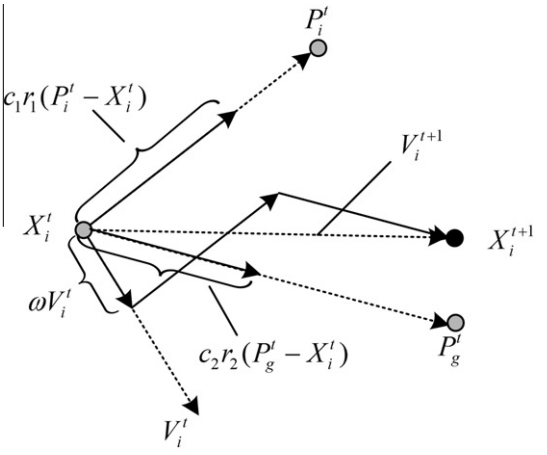

Fig. 1. Schematic drawing of position update process of a particle.

```
Standard PSO
1   // Initialization:
2   for i=1 to the swarm size do
3       for j=1 to the number of dimensions do
4           Initialize X_i within the search range of (X_min, X_max) randomly;
5           Initialize V_i within the velocity range of (V_min, V_max) randomly;
6           P_i = X_i;
7       end for
8   end for
9   Evaluate each particle;
10  Identify the best position P_g;
11  // Loop:
12  While (stop criterion is not satisfied || t > max iteration times) do
13          for i=1 to the swarm size do
14              for j=1 to the number of dimensions do
15                  V_i^{t+1} = ωV_i^t + c_1 r_1 (P_i^t − X_i^t) + c_2 r_2 (P_g^t − X_i^t)
16                  X_i^{t+1} = X_i^t + V_i^{t+1}
17              end for
18              Evaluate fitness(X_i^t);
19              if fitness(P_i^t) < fitness(X_i^t) then
20                Update P_i^t ;
21              end if
22              if fitness(P_g^t) < fitness(P_i^t) then
23                Update P_g^t ;
24              end if
25          end for
26  end while
```

Fig. 2. The pseudo-code of standard PSO.

which the coordinate of the corresponding dimension changes. Such a variant of PSO is mainly used to solve combinatorial optimization problems.

Angeline [2] employed the concept of "selection" from evolutionary computation to PSO. In this version, particles with better fitness were reproduced to the next generation to replace bad particles, which could guarantee that the whole swarm improves the average quality of the particles for every generation. Løvbjerg et al. [21] further discussed the integration of strategies of evolutionary computation with PSO, introducing subpopulations to PSO and using breeding probability instead of fitness. The experimental results showed the effectiveness of this method. Higashi and Iba [10] combined the idea of the particle swarm with "mutation" from evolutionary computation to study performance of the traditional velocity and position update rules with the ideas of Gaussian Mutation. Nakano et al. [23] studies PSO based on the concept of tabu search, in this method, particles are divided into two swarms which play roles of intensification and diversification, respectively. Yin et

al. [41] proposed cyber swarm algorithm by introducing scatter search and path relinking as adaptive memory strategies to PSO. Fan and Zahara [7] integrated Nelder–Mead simplex method into PSO, which not only improves the speed of convergence of simplex method, but also enhances search accuracy of PSO. El-Abd and Kamel [6] discussed the hybridization of PSO with estimation of distribution algorithm (EDA), and proposed a cooperative PSO algorithm based on the migration of heterogeneous probabilistic models.

In order to better balance exploitation capability and exploration capability, neighborhood topologies designed for particles are studied. Four neighborhood topologies comprising circles, wheels, stars and random edges were tested in [14], then an intensive study was conducted in [15]. Included in their work were an investigation on the effects of various neighborhood topologies of the swarm and a discussion on the relationship between population structure and performance. Being different from the static neighborhood topologies above, Suganthan [33] proposed a dynamic neighborhood topology. This topology enables the neighborhood of a particle to expand to include all particles with the increase of iteration times. Parsopoulos and Vrahatis [26] designed a unified PSO (UPSO) to provide a unified version for local best topology and global best topology. Li [18] used nich technology to study PSO, and presented a PSO niching algorithm using a ring neighborhood topology. This algorithm could induce stable niching behavior without the need to specify any niching parameters, and the experimental results suggest that this variant of PSO could provide superior and more consistent performance over some existing PSO niching algorithms that require niching parameters. Mendes et al. [22] proposed the fully informed particle swarm (FIPS) to enable all neighbors to contribute for the particle's update. Peram et al. [27] designed a Fitness–Distance-Ratio based PSO (FDR-PSO). In this PSO variant, a new velocity component, using a ratio of fitness and Euclidean distance to determine which particles to connect, was added to the velocity update equation to optimize each dimension of particles.

In [35], Van den Bergh and Engelbrecht proposed the cooperative particle swarm optimizer (CPSO-H), employing cooperative behavior to improve the performance of PSO. In their work, several populations were used for optimizing different fragments of a solution vector. Kennedy and Mendes [15] proposed comprehensive learning particle swarm optimizer (CLPSO). In CLPSO, comprehensive learning strategy is designed to optimize every dimension of solution vectors, and every particle's personal best solution could be learned by every particle with a specified probability. Experimental results show that CLPSO performs well, especially on multimodal problems; however, there are additional parameters to be empirically tuned.

Up to now, PSO has been effectively applied to optimization problems in various areas, such as power system, job shop scheduling, multi-objective optimization, mechanical design, biological information, dynamic optimization, and stock portfolio construction. In [1], the problem of power-system stabilizers was formulated as an optimization problem and PSO was employed to solve it. In [8], PSO was used to solve the economic dispatch problem in good computation efficiency and stable convergence characteristic while satisfying the generator constraints. In [19,30], modifications of PSO have been proposed to better suit for the job shop scheduling problem with good experimental results. In [38], PSO with a preference order scheme was used to identify the best tradeoff of different objectives in a multi-objective optimization problem, which was demonstrated more efficient than Pareto ranking scheme. In [34], time variant inertia and acceleration coefficients were introduced to design PSO for multi-objective optimization problem. In [11], PSO combined with a gradient-based method was applied to design composite beam subject to strength constraints, and the method greatly improved reserve factors which could be useful in the design of helicopter rotor structures. Rasmussen and Krink[28] used PSO to train hidden markov models for the alignment of protein sequences, it could overcome the shortcomings of trapping in the local optimum when using Baum–Welch algorithm. Du and Li [5] proposed a multi-strategy ensemble PSO (MEPSO) to solve dynamic optimization problems. Compared with previous relevant works, MEPSO has a better robustness in parameter settings. Chang and Shi [3] applied a PSO with moving interval windows to optimize investment allocation of the stocks in a portfolio, ensuring the risk control and obtaining a more profitable stock investment portfolio.

3. Cellular particle swarm optimization

When using population-based optimization algorithm to solve complex problems, two of the most crucial factors significantly affect the capability of algorithms: (1) communication structures, and (2) information inheriting and diffusing mechanisms. The two factors could influence the co-operation of the population and the self-adaption of the individual, and further affect the performance of the algorithms.

Ever since the concept of CA was first proposed by Von Neumann and Ulam, it has become a powerful tool to conduct scientific research. There have been an increasing number of researchers and scientists from different fields exploiting the CA in physics, biology, social science, computer science, and so on. Moreover, along with the publication of the monumental book "*A New Kind of Science*" written by Wolfram [39], CA, together with its vitality, application and development, has been comprehensively presented and discussed, which truly attracted global attention to CA.

The popularity of research on CA and its application has two reasons: (1) the implementation of CA is relatively simple; and (2) the research and application of CA does not necessarily call for rigorous mathematical reasoning. And the emergent behavior of CA, whose homogeneously interconnected deterministic finite automata (cells) work synchronously at discrete time steps obeying one common transition function [16], turned out that a large array of not very powerful elements operating in parallel can be programmed to be very powerful – each cell interacts locally with its neighbors, and then a collection of such cells congregate to present enormous power to solve problems. With this emergent behavior, CA could play an

important role in modeling complex systems. In view of the above, we employed the idea of CA to study swarm system, and thought about PSO in CA way.

3.1. Cellular automata way

Let us consider that a set of cells are distributed in checkerboard-like lattice, each cell is tightly connected with its neighboring cells. Each cell has different states which could be considered as its intrinsic information. At a discrete time step, each cell communicates with its neighbors, and updates its current state according to its communication. All cells communicate and change states synchronously. According to the concept of CA, four basic components of CA could be summarized as: cell state, cell space, neighborhood, and transition rule. Cell state is the number of distinct states that a single cell can be in. A cell space describes how cells are connected with each other. In two-dimensional CA, a lattice structure is used to represent the cell space. Because we usually simulate real-world dynamic systems by finite lattice grids, we need to define the boundary condition. And we introduce periodic boundary by means of imaging the lattice grids embedded in toroidal surface topology, that is, the left boundary connects to the right boundary, and the upper boundary connects to the lower boundary. Neighborhood is a set of cells surrounding a given cell. It affects the next state of the given cell (e.g. Moore neighborhood, which is a set of cells surrounding a given cell with the radius equaling to 1). Transition rule is the control component in CA, it is used to determines the next state of a cell according to its current state and the states of the cells in its neighborhood. So CA could be described briefly as "In a preassigned cell space, every cell updates its current cell state governed by a transition rule according to the state of cells in the neighborhood at discrete time steps."

Although CA and PSO come from different fields, comparing the two mechanisms, we find there are several similar features. Firstly, both of them are comprised of a set of individuals, which are called cells in CA, and particles, in PSO. Secondly, every individual interacts with each other to transmit certain intrinsic information. In CA, each cell communicates with its neighbors and changes its cell state taking into account the cell's current state and its neighbors' states. Similarly, each particle communicates with other particles and updates such intrinsic information as velocity, position, fitness, personal best position, global best position in PSO. Thirdly, in CA, a transition rule is used to govern the evolution of cells, while we use two equations about velocity and position to update particles. Finally, CA and PSO both run in discrete time step.

PSO mimics a swarm system that involves interaction between different particles in the swarm. In order to enhance the performance of the interaction, we focus on employing the idea of CA to explore the communication structure and information inheriting and diffusing mechanisms of the swarm system of PSO. So in the proposed CPSO, we analyze PSO with a CA model, where individuals can only exchange information within their neighborhood. It could help exploring the search space due to the slow information diffusion through the population, promoting the preservation of diversity, and exploiting every cell's local information inside the neighborhood. Note that when constructing the CA model for CPSO, instead of rigorously and rigidly combining the concepts and methods of CA and PSO, we design a mechanism sophisticatedly by integrating adapted CA and PSO so that concertation can be attained. The CA model for PSO is defined as follows:

(a) cell: selected candidate solution (we assume there are U cells);
(b) cell space: the set of all cells;
(c) cell state: the particle's information of such respects as P_i, P_g and X_i at time t, denoted by S_i^t $(i = 1, 2, \ldots, U)$. We let $S_i^t = \left[P_i^t, P_g^t, V_i^t, X_i^t, \ldots\right]$;
(d) neighborhood: a kind of topology based on lattice structure containing a set of particles, defined by $N(i) = \{i + \delta_1, i + \delta_2, \ldots, i + \delta_m, \ldots, i + \delta_l\}$ where l is the size of the set, $m = 1, 2, \ldots, l$;
(e) transition rule: $S_i^{t+1} = f\left(S_{N(i)}^t\right) = f\left(S_i^t, S_{i+\delta_1}^t, S_{i+\delta_2}^t, \ldots, S_{i+\delta_l}^t\right)$;
(f) discrete time step: iterations in PSO.

Based on above, the general framework of CPSO is given in Fig. 3.

Under the framework, two versions of CPSO (CPSO-inner and CPSO-outer) are designed and discussed in the following parts in detail.

The general framework of Cellular Particle Swarm Optimization

```
Initialize swarm size and PSO parameters
Initialize particles' positions and velocities
Loop
        Identify every particle's state
        Implement cellular automata mechanism
        Update velocities
        Update positions
        Exit loop if criterion is met
End loop
```

Fig. 3. The general framework of CPSO.

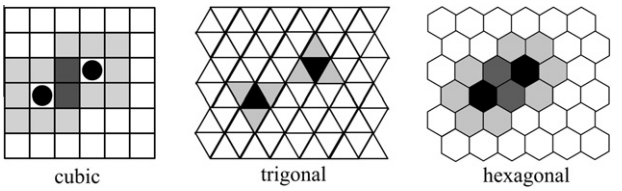

Fig. 4. Three typical lattice structures of CA.

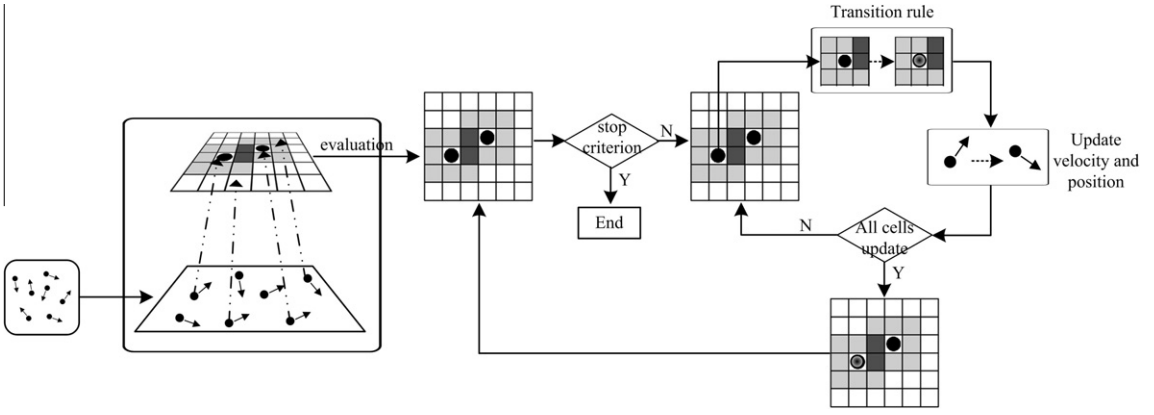

Fig. 5. The flow chart of CPSO-inner.

3.1.1. CPSO-inner

In CPSO-inner, all information is derived and inherited inside the swarm. With a similar way of derivation and inheritation of information, three typical lattice structures are used to construct cell space. Fig. 4 shows three typical lattice structures of CA, namely cubic, trigonal and hexagonal structure, respectively.

We define a particle as a cell, and the set of all cells includes and only includes particles of the swarm in CPSO-inner. According to the swarm size, lattice structure containing the same number of grids with that of the particles is created to hold particles. In the initialization stage, particles are randomly generated; subsequently, each particle is randomly assigned to one unduplicated grid of the lattice structure one by one. Note that the assignment of each particle to the corresponding grid of the lattice structure remains unchanged during all iterations. Neighborhood could be defined on basis of the lattice structures. Taking cubic lattice structure in Fig. 4 as an example, we first assume the swarm size is 36, then 36 particles are randomly generated in the initialization stage, then 36 grids are created to assemble a 6×6 checkerboard-like cubic lattice structure. Each particle is assigned to each grid, and in the following iterations, the assignment of each particle remains unchanged. The gray grids surrounding two black cells are their neighbors, respectively, and deep gray grids represent the overlapping neighbors belonging to two cells. The communication between cells is limited locally to neighborhood, which could help each cell to conduct exploitation inside their neighbors. Meanwhile, the overlapping neighbors provide a link for different neighborhoods, and they could enable each cell's information to diffuse to the whole swarm, which is favorable for exploring the search space.

In CPSO-inner, we define two cell states: (1) the state of personal best position P_i^t, and (2) the state of neighbors' best position P_n^t. And the two kinds of information are denoted by $S_i^t(P_i)$ and $S_i^t(P_n)$, respectively. For simplification, we denote $S_i^t = [P_i^t, P_n^t]$. The transition rule is defined as:

$$\begin{aligned} S_i^{t+1}(P_n) &= f(S_i^t(P_i), S_{i+\delta_1}^t(P_{i+\delta_1}), S_{i+\delta_2}^t(P_{i+\delta_2}), \ldots, S_{i+\delta_l}^t(P_{i+\delta_l})) \\ &= \min\left(fitness(S_i^t(P_i)), fitness\left(S_{i+\delta_1}^t(P_{i+\delta_1})\right), fitness(S_{i+\delta_2}^t(P_{i+\delta_2})), \ldots, fitness(S_{i+\delta_l}^t(P_{i+\delta_l}))\right) \end{aligned} \quad (4)$$

CPSO-inner is a kind of local-best model. Eq. (4) means that the neighbor with best fitness value is chosen for updating the cell state. When using Eq. (3) to integrate CA with PSO, linearly decreasing inertia weight of PSO is adopted to better balance the global search and local search, so we use Eq. (5) in CPSO-inner:

$$V_i^{t+1} = \omega^t V_i^t + c_1 r_1 (S_i^t(P_i) - X_i^t) + c_2 r_2 (S_i^t(P_n) - X_i^t) \quad (5)$$

where $\omega^t = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{T} \times t$, and T is the maximum iteration times. Fig. 5 is the schematic drawing that illustrates the flow chart of CPSO-inner. And the pseudo-code is outlined in Fig. 6.

CPSO-inner

Choose lattice structure (cubic, trigonal or hexagonal);
Swarmsize = the number of grids created based on the chosen lattice structure ;
// Initialization:
for i=1 **to** the swarm size **do**
    Initialize X_i within the search range of (X_{min}, X_{max}) randomly;
    Initialize V_i within the velocity range of (V_{min}, V_{max}) randomly;
    $P_i = X_i$;
end for
Evaluate each particle;
Identify each particle's state S_i;
//Loop:
While (stop criterion is not satisfied || t < max iteration times) **do**
    for i=1 **to** swarmsize **do**
        $V_i^{t+1} = \omega^t V_i^t + c_1 r_1 (S_i^t(P_i) - X_i^t) + c_2 r_2 (S_i^t(P_n) - X_i^t)$
        $X_i^{t+1} = X_i^t + V_i^{t+1}$
        Evaluate $fitness(X_i)$;
        if $fitness(S_i^t(P_i)) < fitness(X_i)$ **then**
            Update $S_i^t(P_i)$;
        end if
        for k=1 **to** the number of neighbors(l) **do**
            if $fitness(S_i^t(P_n)) < fitness(S_{i+\delta_k}^t(P_{i+\delta_k}))$ **then**
                Update $S_i^t(P_n)$;
            end if
        end for
    end for
end while

Fig. 6. The pseudo-code of CPSO-inner.

Different lattice structures represent different topologies of swarm and could lead to different patterns of information interaction. Generally, the research on neighborhood topologies could be classified as variants of CPSO under the general framework of CPSO shown in Fig. 3, because they all focus on the interaction between every single particle – how a set of particles in the neighborhood of a certain particle work on this particle, and how a particle affects the whole swarm system.

3.1.2. CPSO-outer

When a particle moves in the search space, it oscillates due to two external forces exerted by both the personal best position and the global best position. Moreover, each particle could only move following its previous trace, generating a trajectory of damped sinusoidal waves [27]. In order to improve the searching capability, particles should be endued with more talent to free themselves from original trajectories to explore more potential search space, so we let particles cast their eyes on solutions not belonging to the swarm. This idea has led to the proposed CPSO-outer.

In CPSO-outer, we extend a generalized-CA strategy to guide particles in order to enhance their optimization capability. Firstly, we assume that the whole search space is considered as the cell space, and we define every potential candidate solution in the search space as a cell on the basis of the above assumption. To better understand the interaction behavior of cells, we imagine that the cell space is cut by infinite virtual grids, and the cell space could not be subdivided any more. Every grid contains only one solution. In order to differentiate particles and potential candidate solutions, we define a "smart-cell" as a cell that undergoes search behavior, i.e. a "smart-cell" represents a "particle" of PSO. Then, a cell that is not "smart" is one that represents a candidate solution that is not sampled by any of the particles in the search space. Every smart-cell could construct its neighborhood smartly by a neighborhood function (as depicted in the following). The force of neighbors may draw the particle out of its previous trajectory. We give an illustration in Fig. 7 for understanding the mechanism visually. In Fig. 7, we assume that the search space is a plane cut by the range of two variables. All candidate solutions are distributed in the plane, and then we imagine that a set of virtual grids being constructed to cut the search space, and that every grid contains one potential candidate solution. Among these grids, smart-cells are marked by black dots. The ith particle's position X_i^t is defined as the cell state of this particle. When implementing CPSO-outer, only smart-cells participate in updating process.

A smart-cell constructs its neighborhood by a neighborhood function given in Eq. (6).

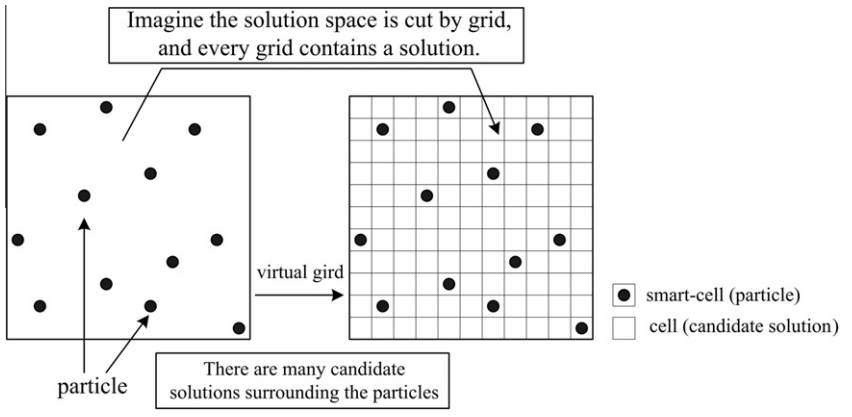

Fig. 7. The schematic drawing for CPSO-outer.

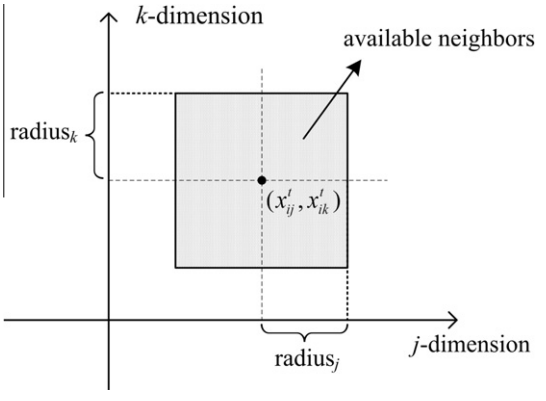

Fig. 8. Neighborhood for j-dimension and k-dimension of $\xi(t)$.

$$N(i) = \begin{cases} X_i^t + \frac{fitness(P_g^t)}{fitness(X_i^t)} R_3 \circ V_i^t & fitness(X_i^t) \neq 0, fitness(P_g^t) \geqslant 0 \\ X_i^t + \left|\frac{fitness(X_i^t)}{fitness(P_g^t)}\right| R_3 \circ V_i^t & fitness(X_i^t) \neq 0, fitness(P_g^t) < 0 \\ X_i^t + \left(\frac{e^{fitness(P_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_3 \circ V_i^t & fitness(X_i^t) = 0, fitness(P_g^t) \geqslant 0 \\ X_i^t + \left(\frac{e^{fitness(P_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_3 \circ V_i^t & fitness(X_i^t) = 0, fitness(P_g^t) < 0 \end{cases} \quad (6)$$

where R_3 is a $1 \times d$ matrix composed by d uniform random numbers in $[-1,1]$, and "$\circ$" is the operation symbol of Hadamard product. We name R_3 as direction coefficient to adjust the direction and distance of the neighborhood. For the jth dimension ($j = 1,2,\ldots,d$) of X_i^t, $N(i)$ generates random points within $\text{radius}_j = \left\|\frac{fitness(P_g^t)}{fitness(X_i^t)} R_{3j} v_{ij}^t\right\|$, $\left\|\frac{fitness(X_i^t)}{fitness(P_g^t)} R_{3j} v_{ij}^t\right\|$ or $\left\|\left(\frac{e^{fitness(P_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_{3j} v_{ij}^t\right\|$ away from x_{ij}^t. We take Fig. 8 as an intuitive description for the neighborhoods of the jth dimension and the kth dimension. l neighbors could be obtained by l different R_3, namely, $R_3^{\delta_m},\ m = 1,2,\ldots,l$. The neighborhood function makes CPSO-outer differ from CPSO-inner and from most variants of PSO adopting static neighbors. Particles in CPSO-outer could select neighbors independently and dynamically to communicate with according to the evaluation of its current position and global best position, and then conduct a local search around every smart-cell. Eq. (6) could be simplified as $N(i) = X_i^t + \xi^t \circ V_i^t$. The range of ξ_i^t could be small at early iterations when the difference of the fitness value of particle with that of the *gbest* is relatively large. Then, along with the particles gradually converging to the *gbest*, that difference reduces and leads to a relatively larger range of ξ_i^t, and finally, when all particles converge to the equilibrium point, the range would have uniform distribution in $[-1,1]$.

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

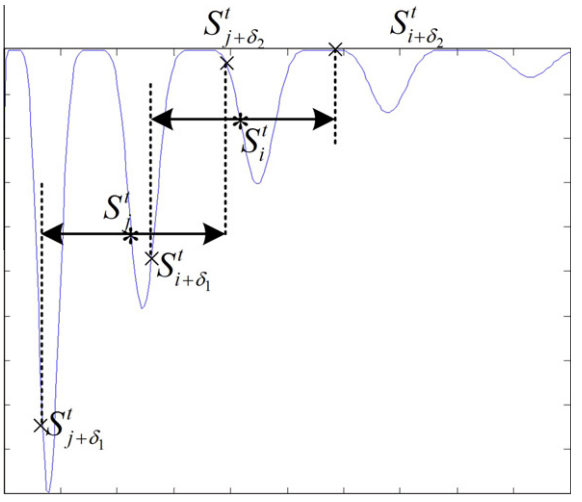

Fig. 9. An illustrative explanation for the transition rule of CPSO-outer.

The transition rule of CA in CPSO-outer is designed as follows:

$$f(\phi) = \min(\mathit{fitness}(N(i)), \mathit{fitness}(N(i+\delta_1)), \ldots, \mathit{fitness}(N(i+\delta_m)), \ldots, \mathit{fitness}(N(i+\delta_l))) \tag{7}$$

where $\phi = \begin{cases} i & \text{if } f(\phi) = \mathit{fitness}(N(i)), \\ i+\delta_m & \text{if } f(\phi) = \mathit{fitness}(N(i+\delta_m)), \end{cases}$

$$s_i^{t+1} = s_\phi^t \tag{8}$$

Eq. (7) means that l neighbors of the ith particle generated by Eq. (6) are evaluated, and the neighbor with the best fitness value is chosen to replace the ith particle. The transition rule could give particles the power to make a wise jump, and it could help exploring the search space in a local competition neighborhood and enhance the diversity of the swarm. So CPSO-outer could have greater potential to search for global optimum in the search space. For further illustration, a two-dimensional function is taken to illustrate conveniently in Fig. 9. We assume there are two particles with the cell state S_i^t and S_j^t, respectively. Every particle could generate two neighbors impacted by $R_3^{\delta_m}$, the ith cell generates two neighbors with the cell state $S^t_{i+\delta_1}$ and $S^t_{i+\delta_2}$. According to the transition rule, $S_i^{t+1} = S^t_{i+\delta_1}$. So the transition rule makes the particle jumps from a local optimal area to another local optimal area with better fitness value. As a result, $S_j^{t+1} = S^t_{j+\delta_1}$, and the jth particle jumps from a local optimal area to the global optimal area.

Compared with CPSO-inner, as well as such variants as CLPSO [20], FIPS [22], and standard PSO, which limit particles interacting inside the swarm, information interacted in CPSO-outer is not only obtained from the swarm, in addition, it considers cells outside the swarm as an important source of information. As has been mentioned, CPSO-inner explores and exploits information only inside the swarm; CLPSO lets particles learn from itself or another particle's personal best in a certain probability; and in the swarm of FIPS, interactions involves all particles. Considering the valuable information in the search space that is not sampled by the particle swarm at each iteration, these PSO variants may suffer from weakening and missing crucial optimization information. In this aspect, CPSO-outer is different and can be superior.

According to the above, the pseudo-code of CPSO-outer is given in Fig. 10.

3.2. Theoretical analysis

In this section, an analysis of the convergence of CPSO is provided. Due to different cellular automata mechanisms used in CPSO-inner and CPSO-outer, respectively, two independent analyses are conducted below.

3.2.1. Analysis for CPSO-inner

To analyze the convergence of CPSO-inner, we are to look at the convergence of PSO first. Van den Bergh and Engelbrecht [36] have studied the convergence of PSO with the inertia weight. Based on their work, an analysis of CPSO-inner is proposed.

At each iteration, the CPSO-inner first updates the swarm following the two equations:

$$V_i^{t+1} = \omega^t V_i^t + c_1 r_1 \left(S_i^t(P_i) - X_i^t\right) + c_2 r_2 \left(S_i^t(P_n) - X_i^t\right) \tag{9}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{10}$$

Then, deriving from Eqs. (9) and (10), we have a second-order difference equations as $\forall i,j$:

$$x_i(t+1,j) - (1 + \omega(t) - c_1 r_1 - c_2 r_2) x_i(t+1,j) + \omega(t) x_i(t-1,j) = c_1 r_1 p_i(t,j) + c_2 r_2 l_i(t,j) \tag{11}$$

```
CPSO-outer
// Initialization:
for i=1 to the swarm size do
    Initialize X_i within the search range of (X_min, X_max) randomly;
    Initialize V_i within the velocity range of (V_min, V_max) randomly;
    P_i = X_i;
end for
Evaluate each particle;
Identify the best position P_g;
Let initialized position be each particle's state;
//Loop:
While (stop criterion is not satisfied || t < maximum iteration times) do
    for i=1 to the swarm size do
        V_i^{t+1} = ω^t V_i^t + c_1 r_1 (P_i^t − X_i^t) + c_2 r_2 (P_g^t − X_i^t)
        X_i^{t+1} = X_i^t + V_i^{t+1}
        Evaluate fitness(X_i^t);
        Generate l neighbors using Eq. (6);
        Evaluate l neighbors;
        for k=1 to the l do
            if fitness(S_i^t) < fitness(S_{i+δ_k}^t) then
                Update S_i^t ;
            end if
        end for
        if fitness(P_i^t) < fitness(S_i^t) then
            Update P_i^t ;
        end if
        if fitness(P_g^t) < fitness(P_i^t) then
            Update P_g^t ;
        end if
    end for
end while
```

Fig. 10. The pseudo-code of CPSO-outer.

Note that $x_i(t,j)$, $p_i(t,j)$, $l_i(t,j)$ are the jth dimension of *particle* i, *pbest* i and *lbest* i, respectively, at iteration t for simplification. Denote $\omega^t = \omega(t)$ for clarity.

The characteristic equation of Eq. (11) is:

$$\lambda^2 - (1 + \omega(t) - c_1 r_1 - c_2 r_2)\lambda + \omega(t) = 0 \tag{12}$$

The eigenvalues are: $\forall i,j$:

$$\lambda_1 = \frac{(1 + \omega(t) - c_1 r_1 - c_2 r_2) + \sqrt{(1 + \omega(t) - c_1 r_1 - c_2 r_2)^2 - 4\omega(t)}}{2} \tag{13}$$

$$\lambda_2 = \frac{(1 + \omega(t) - c_1 r_1 - c_2 r_2) - \sqrt{(1 + \omega(t) - c_1 r_1 - c_2 r_2)^2 - 4\omega(t)}}{2} \tag{14}$$

With initial conditions as $x_i(0,j)$ and $x_i(1,j)$, the general solution $X_i^t\prime$ and a special solution X_i^t* to Eq. (11) can be derived:

$$\forall i,j : x_i'(t,j) = A\lambda_1^t + B\lambda_2^t \tag{15}$$

$$\forall i,j : x_i^*(t,j) = \frac{c_1 r_1 p_i(t,j) + c_2 r_2 l_i(t,j)}{c_1 r_1 + c_2 r_2} \tag{16}$$

where $\forall i,j$:

$$A = -\frac{x_i(2,j) - x_i(1,j)(1 + \lambda_2) + x_i(0,j)\lambda_2}{\lambda_1 - \lambda_2 + \lambda_1\lambda_2 - \lambda_1^2} \tag{17}$$

$$B = -\frac{x_i(2,j) - x_i(1,j)(1 + \lambda_1) + x_i(0,j)\lambda_1}{\lambda_2 - \lambda_1 + \lambda_1\lambda_2 - \lambda_2^2} \tag{18}$$

and $\forall i,j : x(2,j) = (1+\omega(1)-c_1r_1-c_2r_2)x_i(1,j) - \omega(1)x_i(0,j) + c_1r_1p_i(1,j) + c_2r_2l_i(1,j)$. Then the solution to Eq. (11) is the sum of the general solution and the special solution:

$$\forall i,j : x_i(t,j) = A\lambda_1^t + B\lambda_2^t + \frac{c_1r_1p_i(t,j) + c_2r_2l_i(t,j)}{c_1r_1+c_2r_2} \tag{19}$$

According to [36], a sufficient and necessary condition to the convergence of PSO is: $\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = 0$.
If the parameters of the algorithm satisfy the following relation:

$$(1+\omega(t)-c_1r_1-c_2r_2)^2 - 4\omega(t) \geqslant 0 \tag{20}$$

Then, the eigenvalues are real, and as described in [36], when $\omega(t)$ is constant, a sufficient and necessary condition to the convergence of the algorithm is:

$$\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1 \tag{21}$$

If the parameters of the algorithm do not satisfy the Inequality (21), which means

$$(1+\omega(t)-c_1r_1-c_2r_2)^2 - 4\omega(t) < 0 \tag{22}$$

Then, the eigenvalues are complex,

$$\lambda_1 = R(\cos\theta + i\sin\theta) \tag{23}$$
$$\lambda_2 = R(\cos\theta - i\sin\theta) \tag{24}$$

where

$$R = \sqrt{\omega(t)} \tag{25}$$
$$\theta = \arctan\frac{\sqrt{4\omega(t) - (1+\omega(t)-c_1r_1-c_2r_2)^2}}{1+\omega(t)-c_1r_1-c_2r_2} \tag{26}$$
$$A\lambda_1^t + B\lambda_2^t = R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] \tag{27}$$

When $\omega(t)$ is constant, a sufficient and necessary condition to the convergence of the algorithm is $\|R\| < 1$, which means max $\{\|\lambda_1\|, \|\lambda_2\|\} < 1$, so that:

$$\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = \lim_{t\to\infty} R^t[(A+B)\cos(t\theta) + \mathrm{i}(A-B)\sin(t\theta)] = 0 \tag{28}$$

Therefore, $\forall i,j$:

$$\lim_{t\to\infty} x_i(t,j) = \lim_{t\to\infty}\left(A\lambda_1^t + B\lambda_2^t + \frac{c_1r_1p_i(t,j) + c_2r_2l_i(t,j)}{c_1r_1+c_2r_2}\right) = \lim_{t\to\infty}\frac{c_1r_1p_i(t,j) + c_2r_2l_i(t,j)}{c_1r_1+c_2r_2} \tag{29}$$

Then,

$$\lim_{t\to\infty} X_i^t = \lim_{t\to\infty}\frac{c_1r_1S_i^t(P_i) + c_2r_2S_i^t(P_n)}{c_1r_1+c_2r_2} \tag{30}$$

According to Eq. (30), in infinite iterations, CPSO-inner becomes a search on a point W that satisfies the relation:

$$\forall i : W = \kappa\cdot S_i^t(P_i) + (1-\kappa)\cdot S_i^t(P_n) \tag{31}$$

where $\kappa = \frac{c_1r_1}{c_1r_1+c_2r_2}$,and $\kappa \in [0,1]$.

Denote $\pi = \{W|W = \kappa S_i^t(P_i) + (1-\kappa)S_i^t(P_n),\ 0 \leqslant \kappa \leqslant 1\}$, and assume $\forall A \in \pi$, then $\exists\kappa_A : W = \kappa_A\cdot S_i^t(P_i) + (1-\kappa_A)\cdot S_i^t(P_n)$.
Since $\exists r_1, r_2 \in [0,1] : \frac{c_1r_1}{c_1r_1+c_2r_2} = \kappa_A \Longleftrightarrow 1 - \frac{c_1r_1}{c_1r_1+c_2r_2} = 1-\kappa_A \Longleftrightarrow \frac{c_2r_2}{c_1r_1+c_2r_2} = 1-\kappa_A$,

$$\exists r_1, r_2 \in [0,1] : \frac{c_1r_1S_i^t(P_i) + c_2r_2S_i^t(P_n)}{c_1r_1+c_2r_2} = A \tag{32}$$

Denote the probability for $\frac{c_1r_1S_i^t(P_i)+c_2r_2S_i^t(P_n)}{c_1r_1+c_2r_2} = A$ to be true as $\rho_i^t(A)$. Therefore, $\forall A \in \pi : \rho_i^t(A) > 0$.

Since $S_i^t(P_n) \in \pi$, we have $\exists B \in \pi : f(B) = f(S_i^t(P_n))$.

Denote $\gamma = \{B|B \in \pi, f(B) = f(S_i^t(P_n))\}$, then $\gamma \cap \pi \neq \phi$.

Given $\exists B: B \in \gamma \cap \pi$, then, $\forall t, \exists m : 0 < \rho_i^t(B) < 1$.

Therefore, *prob*, the possibility for $\exists t : \frac{c_1r_1S_i^t(P_i)+c_2r_2S_i^t(P_n)}{c_1r_1+c_2r_2} = B$ in t' number of iterations is:

$$prob = 1 - \prod_{t=1}^{t'}(1-\rho_i^t(B)) \tag{33}$$

Then,

$$\lim_{t'\to\infty} prob = \lim_{t'\to\infty}\left[1-\prod_{t=1}^{t'}(1-\rho_i^t(B))\right] = 1 \tag{34}$$

This is to say, in an infinite number of iterations, the probability for $\exists t : \frac{c_1 r_1 S_i^t(P_i)+c_2 r_2 S_i^t(P_n)}{c_1 r_1 + c_2 r_2} = B$ to be true is 1.

Since *pbest* records the best solution the corresponding particle has ever reached, and, *lbest* records the best solution that the corresponding particle or any of its neighbors has ever reached, therefore, $f(S_i^t(P_i)) \leqslant f(S_i^t(P_n)) = f(B)$, and when a particle reaches B, the corresponding *pbest* is to record this position.

Therefore, $\exists B \in \pi : \lim_{t\to\infty} P_i^t = B$, which means that

$$\forall i : \lim_{t\to\infty} f(S_i^t(P_i)) = f(S_i^t(P_n)) \tag{35}$$

According to Eq. (35),

$$\forall i : \lim_{t\to\infty} f(S_i^t(P_i)) = \lim_{t\to\infty} f(S_i^t(P_n)) \tag{36}$$

Then we take into consideration the mechanism of CA under the framework of CPSO-inner. For the convenient discussion, in the following, $Cell_i^t$ denotes the ith cell in the lattice structure, $S_i^t(P_i)$ and $S_i^t(P_n)$ denote the two corresponding states of $Cell_i^t$, *pbest i* and *lbest i*, respectively. $N\left(Cell_i^t\right)$ denotes the set of the neighbors of $Cell_i^t$. $N(S_i^t(P_i))$ and $N(S_i^t(P_n))$ denote the set of the neighbor's *pbests* and *lbests*, respectively. CA strategy of CPSO-inner has the following definitions and properties:

Definition 1. For $i,j \in (1,2,\ldots,N)$, a relation which satisfies the following:

(i) $Cell_i^t \in N\left(Cell_j^t\right)$,

(ii) $Cell_j^t \in N\left(Cell_i^t\right)$

is defined as a *link*, denoted as $link\left(Cell_i^t, Cell_j^t\right)$ or $link\left(Cell_j^t, Cell_i^t\right)$. For simplification, we say that $Cell_i^t$ and $Cell_j^t$ are connected by a *link*.

Property 1. *In the lattice of CA, if* $Cell_i^t \in N\left(Cell_j^t\right)$, *then,* $Cell_j^t \in N\left(Cell_i^t\right)$.

Property 2. *Any two cells in the lattice can be connected by a finite numbers of links, given the scale of the lattice (number of the cells in the lattice) is finite.*

Property 3.

$$S_i^t(P_n) = choose\left(\{S_i^t(P_i)\} \bigcup N(S_i^t(P_i))\right) \tag{37}$$

where operator choose(A) outputs the element with the best fitness value in the Set A's members.

Properties 1–3 are intuitive on our structure of lattice, and they are characteristics of cellular automata. Therefore, we do not provide the proofs for them.

Denote g^t as the best fitness value CPSO-inner has ever reached at iteration t, and g is the best fitness value CPSO-inner can reach at a certain run, namely, $\lim_{t\to\infty} g^t = g^*$. Denote $\beta = \{G_1, G_2, \ldots, G_k, \ldots, G_u\}$ as the set containing all the points with g as the objective value among the search space.

Property 4. $\forall t, \exists i : g^t = f(S_i^t(P_i)) = f(S_i^t(P_n))$.

Property 5. *$\forall i$: if, at iteration t,* $f(S_i^t(P_i)) = g^*$, *then* $\forall I \in N(S_i^t(P_n)) : f(I) = g^*$.

Property 6. *$\forall i$: if, at iteration t,* $\exists I \in N(S_i^t(P_i)) : f(I) = g^*$, *then,* $f(S_i^t(P_n)) = g^*$.

Properties 4–6 can be easily derived from Property 3.

Property 7. *$\forall i$: if, at iteration t,* $f(S_i^t(P_i)) = g^*$, *then* $\forall J \in N(S_i^t(P_i)) : \lim_{t\to\infty} f(J) = g^*$.

Proof. With assumptions of Property 7, deriving from Property 5, $\forall I \in N(S_i^t(P_n)) : f(I) = g^*$. Then, in accordance with Eq. (36), $\exists J \in N(S_i^t(P_i)) : \lim_{t\to\infty} f(J) = \lim_{t\to\infty} f(I) = \lim_{t\to\infty} g^* = g^*$. □

Therefore, Property 7 is true.

Property 8. $\forall i$: *if, at iteration t,* $\exists I \in N(S_i^t(P_i)) : f(I) = g^*$, *then,* $\lim_{t\to\infty} f(S_i^t(P_i)) = g^*$.

Due to Property 6, under the assumption of Property 8, $f\left(S_i^{t+1}(P_n)\right) = g^*$ is true. Then, due to Eq. (36) $\lim_{t\to\infty} f\left(S_i^{t+1}(P_i)\right) = \lim_{t\to\infty} f\left(S_i^{t+1}(P_n)\right) = \lim_{t\to\infty} g^* = g^*$.

Therefore, Property 8 is true.

According to Property 4,

$$\exists i : \lim_{t\to\infty} f(S_i^t(P_i)) = \lim_{t\to\infty} f(S_i^t(P_n)) = \lim_{t\to\infty} g^t = g^* \tag{38}$$

Then, according to Properties 2, 7 and 8,

$$\forall i : \lim_{t\to\infty} f(S_i^t(P_i)) = \lim_{t\to\infty} f(S_i^t(P_n)) = g^* \tag{39}$$

so that CA comes to a equilibrium state.

Then, according to Eqs. (30) and (36), $\forall i$:$\exists M(i)$, $K(i) \in \{1,2,\ldots,u\}$ so that

$$\lim_{t\to\infty} X_i = \lim_{t\to\infty} \frac{c_1 r_1 S_i^t(P_i) + c_2 r_2 S_i^t(P_n)}{c_1 r_1 + c_2 r_2} = \frac{c_1 r_1 G_{M(i)} + c_2 r_2 G_{K(i)}}{c_1 r_1 + c_2 r_2} \tag{40}$$

If $M(i) = K(i)$, then $\lim_{t\to\infty} X_i(t,j) = G_{M(i)} = G_{K(i)}$;

Otherwise, if $M(i) \neq K(i)$, since $E[c_1 r_1] = \frac{1}{2}c_1$, and $E[c_2 r_2] = \frac{1}{2}c_2$, we have $\lim_{t\to\infty} X_i^t = \frac{c_1 G_{M(i)} + c_2 G_{K(i)}}{c_1 + c_2}$.

In sum, the equilibrium point of a particle of CPSO-inner is:$\forall i$:

$$\lim_{t\to\infty} X_i^t = \begin{cases} G_{M(i)} & M(i) = K(i) \\ \frac{c_1 G_{M(i)} + c_2 G_{K(i)}}{c_1 + c_2} & M(i) \neq K(i) \end{cases} \tag{41}$$

where $f(G_{M(i)}) = f(G_{K(i)}) = g^*$

Note that in the above analysis, we assume that $\omega(t)$ is constant. However, in a number of variants of PSO in literatures and in our proposed CPSO-inner, $\omega(t)$ changes with iterations. In this case, the CPSO-inner will converge, if $\exists T$, where T is finite, so that, when $t > T$, $\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1$.

Proof. When $(1 + \omega(t) - c_1 r_1 - c_2 r_2)^2 - 4\omega(t) \geqslant 0$

$$\lim_{t\to\infty} \left(A\lambda_1^t + B\lambda_2^t\right) = \lim_{t\to\infty} \left(A\lambda_1^T \lambda_1^{t-T} + B\lambda_2^T \lambda_1^{t-T}\right) \tag{42}$$

since $\exists Q$, where Q is finite, so that $-Q < \lambda_1^T < Q$ and $-Q < \lambda_2^T < Q$.

$$\lim_{t\to\infty} \left(A\lambda_1^t + B\lambda_2^t\right) = \lim_{t\to\infty} \left(A\lambda_1^T \cdot 0 + B\lambda_2^T \cdot 0\right) = 0 \tag{43}$$

When $(1 + \omega(t) - c_1 r_1 - c_2 r_2)^2 - 4\omega(t) < 0$

$$\lim_{t\to\infty} \left(A\lambda_1^t + B\lambda_2^t\right) = \lim_{t\to\infty} R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] = \lim_{t\to\infty} R^T R^{t-T}[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] \tag{44}$$

Since $\exists Q$, where Q is finite, so that $-Q < R^T < Q$. When $t > T$, $\max\{\|\lambda_1\|, \|\lambda_2\|\} = R < 1$

$$\begin{aligned} \lim_{t\to\infty} \left(A\lambda_1^t + B\lambda_2^t\right) &= \lim_{t\to\infty} R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] = \lim_{t\to\infty} R^T \cdot R^{t-T} \cdot [(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] \\ &= \lim_{t\to\infty} R^T \cdot 0 \cdot [(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] \end{aligned}$$

Therefore, CPSO-inner will converge, if $\exists T$, where T is finite, so that, when $t > T$, $\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1$, in the case that $\omega(t)$ changes with iterations. □

3.2.2. Analysis for CPSO-outer

In this section, an analysis of the convergence of CPSO-outer is provided, and the behavior of the particles is revealed. Van den Bergh and Engelbrecht [36] analyzed particle trajectory and presented that each particle in the swarm of canonical PSO with the inertia weight converges to a stable point. Following their work, an analysis of the behavior of the particles in CPSO-outer is proposed.

At each iteration, the CPSO-outer updates the swarm following the three equations:

$$V_i^{t+1} = \omega^t V_i^t + c_1 r_1 \left(P_i^t - S_i^t\right) + c_2 r_2 \left(P_g^t - S_i^t\right) \tag{45}$$

$$X_i^{t+1} = S_i^t + V_i^{t+1} \tag{46}$$

$$S_i^{t+1} = choose\left(\left\{X_i^{t+1}, X_i^{t+1} + \Delta_1^{t+1} \circ V_i^{t+1}, X_i^{t+1} + \Delta_2^{t+1} \circ V_i^{t+1}, \ldots, X_i^{t+1} + \Delta_m^{t+1} \circ V_i^{t+1}, \ldots, X_i^{t+1} + \Delta_l^{t+1} \circ V_i^{t+1}\right\}\right) \tag{47}$$

where *choose*(A) outputs the element with the best fitness value within Set A, and

$$\xi_m^t = \begin{cases} \frac{fitness(P_g^t)}{fitness(X_i^t)} R_3^{\delta_m}, & fitness(X_i^t) \neq 0, \quad fitness(P_g^t) \geqslant 0 \\ \left|\frac{fitness(X_i^t)}{fitness(P_g^t)}\right| R_3^{\delta_m}, & fitness(X_i^t) \neq 0, \quad fitness(P_g^t) < 0 \\ \left(\frac{e^{fitness(P_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_3^{\delta_m}, & fitness(X_i^t) = 0, \quad fitness(P_g^t) \geqslant 0 \\ \left(\frac{e^{fitness(P_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_3^{\delta_m}, & fitness(X_i^t) = 0, \quad fitness(P_g^t) < 0 \end{cases} \tag{48}$$

The elements of Set $\left\{X_i^{t+1}, X_i^{t+1} + \Delta_1^{t+1} \circ V_i^{t+1}, X_i^{t+1} + \Delta_2^{t+1} \circ V_i^{t+1}, \ldots, X_i^{t+1} + \Delta_m^{t+1} \circ V_i^{t+1}, \ldots, X_i^{t+1} + \Delta_l^{t+1} \circ V_i^{t+1}\right\}$ are candidate solutions generated by CA according to Eq. (6). In Eq. (47), S_i^{t+1} is generated by a generalized-CA strategy, which searches the neighbors of X_i^{t+1}. Substantially, the generalized-CA strategy generates a set of candidate solutions, and assign the one with the best fitness value to S_i^{t+1}. This strategy refines the position of each particle at each iteration, by searching the local area defined by Eq. (6); when there is a better solution attained by CA, the position of the particle is replaced by this better solution. Denote that

$$\xi_i^{t+1} \circ V_i^{t+1} = \left\{X_i^{t+1}, X_i^{t+1} + \Delta_1^{t+1} \circ V_i^{t+1}, X_i^{t+1} + \Delta_2^{t+1} \circ V_i^{t+1}, \ldots, X_i^{t+1} + \Delta_m^{t+1} \circ V_i^{t+1}, \ldots, X_i^{t+1} + \Delta_l^{t+1} \circ V_i^{t+1}\right\} - X_i^{t+1} \tag{49}$$

That is:

$$S_i^{t+1} = X_i^{t+1} + \xi_i^{t+1} \circ V_i^{t+1} \tag{50}$$

Combining Eqs. (45), (46) and (50), we have

$$S_i^{t+1} = S_i^t + \omega^t V_i^t + c_1 r_1 (P_i^t - S_i^t) + c_2 r_2 \left(P_g^t - S_i^t\right) + \xi_i^{t+1} \circ \left(\omega^t V_i^t + c_1 r_1 (P_i^t - S_i^t) + c_2 r_2 \left(P_g^t - S_i^t\right)\right) \tag{51}$$

Note that $x_i(t,j)$, $p_i(t,j)$, $p_g(t,j)$, $\varepsilon_i(t,j)$, $v_i(t,j)$ are the jth dimension of S_i^t, P_i^t, P_g^t, ξ_i^t, and V_i^t, respectively, at iteration t for simplification. Denote $\omega^t = \omega(t)$ for clarity.

According to Eq. (51), we have: $\forall i,j$:

$$\begin{aligned} s_i(t+1,j) = {} & s_i(t,j) + (1+\varepsilon(t+1,j))\omega(t) v_i(t,j) + (1+\varepsilon(t+1,j)) c_1 r_1 (p_i(t,j) - s_i(t,j)) + (1+\varepsilon(t+1,j)) c_2 r_2 (p_g(t,j) \\ & - s_i(t,j)) \end{aligned} \tag{52}$$

$$v_i(t,j) = x_i(t,j) - s_i(t-1,j) \tag{53}$$

$$s_i(t,j) = x_i(t,j) + \varepsilon_i(t,j) v_i(t,j) \tag{54}$$

Combining Eqs. (54) and (55), we have:

$$(1+\varepsilon_i(t,j)) v_i(t,j) = s_i(t,j) - s_i(t-1,j) \tag{55}$$

Combining Eqs. (52) and (55), the recurrence relation of CPSO-outer is:

$$\begin{aligned} (1+\varepsilon(t,j)) s_i(t+1,j) = {} & (1+\varepsilon(t,j)) s_i(t,j) + \omega(t)(1+\varepsilon(t+1,j))(s_i(t,j) - s_i(t-1,j)) + (1+\varepsilon(t,j))(1+\varepsilon(t+1,j)) \\ & \times c_1 r_1 (p_i(t,j) - s_i(t,j)) + (1+\varepsilon(t,j))(1+\varepsilon(t+1,j)) c_2 r_2 (p_g(t,j) - s_i(t,j)) \end{aligned} \tag{56}$$

Denote $\phi_{t+1} = 1 + \varepsilon(t+1,j)$, and thus, $\phi_t = 1 + \varepsilon(t,j)$. Deriving from Eq. (56), the recurrence relation of CPSO-outer becomes a second-order difference equations as follows: $\forall i,j$:

$$\begin{pmatrix} \phi_t \\ -\phi_t - \phi_{t+1}\omega(t) + \phi_t \phi_{t+1}(c_1 r_1 + c_2 r_2) \\ \phi_{t+1}\omega(t) \end{pmatrix}^T \begin{pmatrix} s_i(t+1,j) \\ s_i(t,j) \\ s_i(t-1,j) \end{pmatrix} = \phi_t \phi_{t+1} (c_1 r_1 p_i(t,j) + c_2 r_2 p_g(t,j)) \tag{57}$$

The characteristic equation of Eq. (57) is: $\forall i,j$:

$$\begin{pmatrix} \phi_t \\ -\phi_t - \phi_{t+1}\omega(t) + \phi_t \phi_{t+1}(c_1 r_1 + c_2 r_2) \\ \phi_{t+1}\omega(t) \end{pmatrix}^T \begin{pmatrix} \lambda^2 \\ \lambda \\ 1 \end{pmatrix} = 0 \tag{58}$$

The solutions to this equation, or namely, the eigenvalues are: $\forall i,j$:

$$\lambda_1 = \frac{[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)] + \sqrt{[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)]^2 - 4\phi_t\phi_{t+1}\omega(t)}}{2\phi_t} \tag{59}$$

$$\lambda_1 = \frac{[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)] - \sqrt{[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)]^2 - 4\phi_t\phi_{t+1}\omega(t)}}{2\phi_t} \tag{60}$$

Then, with initial conditions as $s_i(0,j)$ and $s_i(1,j)$, the general solution $s_i'(t,j)$ and a special solution $s_i^*(t,j)$ to Eq. (57) can be derived: $\forall i,j$:

$$s_i'(t,jt) = A\lambda_1^t + B\lambda_2^t \tag{61}$$

$$s_i^*(t,jt) = \frac{c_1r_1p_i(t,j) + c_2r_2p_g(t,j)}{c_1r_1 + c_2r_2} \tag{62}$$

where

$$A = -\frac{s_i(2,j) - s_i(1,j)(1+\lambda_2) + s_i(0,j)\lambda_2}{\lambda_1 - \lambda_2 + \lambda_1\lambda_2 - \lambda_1^2} \tag{63}$$

$$B = -\frac{s_i(2,j) - s_i(1,j)(1+\lambda_1) + s_i(0,j)\lambda_1}{\lambda_2 - \lambda_1 + \lambda_1\lambda_2 - \lambda_1^2} \tag{64}$$

And

$$s_i(2,j) = [1 + (\phi_{t+1}/\phi_t)\omega(1) - \phi_{t+1}(c_1r_1 + c_2r_2)]s_i(1,j) - (\phi_{t+1}/\phi_t)\omega(1)s_i(0,j) + \phi_{t+1}(c_1r_1p_i(1,j) + c_2r_2p_g(1,j)) \tag{65}$$

Then the solution to Eq. (57) is the sum of the general solution and the special solution: $\forall i,j$:

$$S_i^t = A\lambda_1^t + B\lambda_2^t + \frac{c_1r_1p_i(t,j) + c_2r_2p_g(t,j)}{c_1r_1 + c_2r_2} \tag{66}$$

A sufficient and necessary condition to the convergence of PSO is: $\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = 0$.
When the parameters of the algorithm satisfy the following relation:

$$[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)]^2 - 4\phi_t\phi_{t+1}\omega(t) \geqslant 0 \tag{67}$$

the eigenvalues are real, and as described in [36], when $\omega(t)$ is constant, a sufficient and necessary condition to $\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = 0$ is:

$$\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1 \tag{68}$$

So that $\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = 0$.
If the parameters of the algorithm do not satisfy the Inequality (67), which means

$$[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)]^2 - 4\phi_t\phi_{t+1}\omega(t) < 0 \tag{69}$$

Then, the eigenvalues are complex,

$$\lambda_1 = R(\cos\theta + \imath\sin\theta) \tag{70}$$

$$\lambda_2 = R(\cos\theta - \imath\sin\theta) \tag{71}$$

where

$$R = \sqrt{\phi_{t+1}\omega(t)} \tag{72}$$

$$\theta = \arctan\frac{\sqrt{4\phi_t\phi_{t+1}\omega(t) - [\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)]^2}}{\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1r_1 + c_2r_2)} \tag{73}$$

$$A\lambda_1^t + B\lambda_2^t = R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] \tag{74}$$

When $\omega(t)$ is constant, a sufficient and necessary condition to $\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = 0$ is $\|R\| < 1$, which means $\max\{\|\lambda_1\|,\|\lambda_2\|\} < 1$, so that:

$$\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = \lim_{t\to\infty}R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] = 0 \tag{75}$$

Therefore, we have: $\forall i,j$:

$$\lim_{t\to\infty}s_i(t,j) = \lim_{t\to\infty}\left(A\lambda_1^t + B\lambda_2^t + \frac{c_1r_1p_i(t,j) + c_2r_2p_g(t,j)}{c_1r_1 + c_2r_2}\right) = 0 + \lim_{t\to\infty}\frac{c_1r_1p_i(t,j) + c_2r_2p_g(t,j)}{c_1r_1 + c_2r_2} \tag{76}$$

According to Eq. (76),

$$\lim_{t\to\infty} S_i^t = \lim_{t\to\infty}\left(\frac{c_1 r_1 P_i^t + c_2 r_2 P_g^i}{c_1 r_1 + c_2 r_2}\right) \tag{77}$$

Denote the best fitness value that CPSO-outer can attain, at a certain run, as g^*, which means that the best position that any particle has reached is with an fitness value of g^*. So we have $\lim_{t\to\infty} f\left(P_g^t\right) = g^*$. Denote $\beta = \{G_1, G_2, \ldots, G_k, \ldots, G_u\}$ as the set containing all the points with g^* as the fitness value among the search space. Therefore, $\exists K$: $\lim_{t\to\infty} P_g^t = G_K$.

According to Eq. (77), in infinite iterations, CPSO-outer becomes a search on a point W that satisfies the relation:

$$\forall i : W = \kappa P_i^t + (1-\kappa) P_g^t \tag{78}$$

where $\kappa = \frac{c_1 r_1}{c_1 r_1 + c_2 r_2}$,and $\kappa \in [0,1]$.

Denote $\pi = \left\{W | W = \kappa P_i^t + (1-\kappa) P_g^t, 0 \leqslant \kappa \leqslant 1\right\}$, and assume $\forall A \in \pi,\ \exists \kappa_A : A = \kappa_A P_i^t + (1-\kappa_A) P_g^t$. Since $\exists r_1, r_2 \in [0,1] : \frac{c_1 r_1}{c_1 r_1 + c_2 r_2} = \kappa_A \Longleftrightarrow 1 - \frac{c_1 r_1}{c_1 r_1 + c_2 r_2} = 1 - \kappa_A \Longleftrightarrow \frac{c_2 r_2}{c_1 r_1 + c_2 r_2} = 1 - \kappa_A,\ \exists r_1, r_2 \in [0,1] : \frac{c_1 r_1 P_i^t + c_2 r_2 P_g^t}{c_1 r_1 + c_2 r_2} = A$

Denote the probability for $\frac{c_1 r_1 P_i^t + c_2 r_2 P_g^t}{c_1 r_1 + c_2 r_2} = A$ to be true as $\rho_i^t(A)$. Therefore, $\forall A \in \pi : \rho_i^t(A) > 0$.

Since $P_g^t \in \pi$, we have $\exists B \in \pi : f(B) = f\left(P_g^t\right)$. Denote $\gamma = \left\{B | B \in \pi, f(B) = f\left(P_g^t\right)\right\}$, then we have $\gamma \cap \pi \neq \phi$.

Given $\exists B{:}B \in \gamma \cap \pi$. Then, $\forall t : 0 < \rho_i^t(B) < 1$.

Therefore, *prob*, the possibility for $\exists t : \frac{c_1 r_1 P_i^t + c_2 r_2 P_g^t}{c_1 r_1 + c_2 r_2} = B$ in t' number of iterationsis:

$$prob = 1 - \prod_{t=1}^{t'} (1 - \rho_i^t(B)) \tag{79}$$

Then,

$$\lim_{t'\to\infty} prob = \lim_{t'\to\infty}\left[1 - \prod_{t=1}^{t'} (1 - \rho_i^t(B))\right] = 1 \tag{80}$$

This is to say, in an infinite number of iterations, the probability for $\exists t : \frac{c_1 r_1 P_i^t + c_2 r_2 P_g^t}{c_1 r_1 + c_2 r_2} = B$ to be true is 1.

Since *pbest* records the best solution the corresponding particle has ever reached, and, *gbest* records the best solution any particle has ever reached, therefore, $f(P_i^t) \leqslant f(P_g^t) = f(B)$, and when a particle reaches B, the corresponding *pbest* is to record this position.

Therefore, $\exists B \in \pi : \lim_{t\to\infty} P_i^t = B$, which means that $\lim_{t\to\infty} f(P_i^t) = f\left(P_g^t\right)$.

Also, since $\exists k : \lim_{t\to\infty} P_g^t = G_k$, we have $\forall i : \lim_{t\to\infty} f(P_i^t) = f(G_k)$. Therefore, $\forall i, \exists m : \lim_{t\to\infty} P_i^t = G_m$, denote: $\forall i : \lim_{t\to\infty} P_i^t = G_{M(i)}, M(i) \in \{1, 2, \ldots, u\}$.

$\forall i$: If $M(i) = K,\ \lim_{t\to\infty} S_i^t = \lim_{t\to\infty} \frac{c_1 r_1 G_K + c_2 r_2 G_{M(i)}}{c_1 r_1 + c_2 r_2} = G_K = G_{M(i)}$, where $G_K, G_{M(i)} \in \beta$.

Otherwise, $\forall i$: if $M(i) \neq K$, According to Eq. (77), since $E[c_1 r_1] = \frac{1}{2} c_1$, and $E[c_2 r_2] = \frac{1}{2} c_2$, we have $\lim_{t\to\infty} S_i^t = \frac{c_1 G_K + c_2 G_{M(i)}}{c_1 + c_2}$.

In sum, the equilibrium points of particles of CPSO-outer is:

$$\forall i : \lim_{t\to\infty} S_i^t = \begin{cases} G_K & K = M(i) \\ \frac{c_1 G_M(i) + c_2 G_K}{c_1 + c_2} & K \neq M(i) \end{cases} \tag{81}$$

where $f(G_{M(i)}) = f(G_K) = g^*$

Note that in the above analysis, we assume that $\omega(t)$ is constant. However, in a number of variants of PSO in literatures and in our proposed CPSO-outer, $\omega(t)$ changes with iterations. In this case, CPSO-outer will converge, if $\exists T$, where T is finite, so that, when $t > T$, $\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1$.

Proof. When $[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1 r_1 + c_2 r_2)]^2 - 4\phi_t\phi_{t+1}\omega(t) \geqslant 0$

$$\lim_{t\to\infty}\left(A\lambda_1^t + B\lambda_2^t\right) = \lim_{t\to\infty}\left(A\lambda_1^T\lambda_1^{t-T} + B\lambda_2^T\lambda_1^{t-T}\right) \tag{82}$$

since $\exists Q$, where Q is finite, so that $-Q < \lambda_1^T < Q$ and $-Q < \lambda_2^T < Q$.

$$\lim_{t\to\infty}\left(A\lambda_1^t + B\lambda_2^t\right) = \lim_{t\to\infty}\left(A\lambda_1^T \cdot 0 + B\lambda_2^T \cdot 0\right) = 0 \tag{83}$$

When $[\phi_t + \phi_{t+1}\omega(t) - \phi_t\phi_{t+1}(c_1 r_1 + c_2 r_2)]^2 - 4\phi_t\phi_{t+1}\omega(t) < 0$

$$\lim_{t\to\infty}\left(A\lambda_1^t + B\lambda_2^t\right) = \lim_{t\to\infty} R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] = \lim_{t\to\infty} R^T R^{t-T}[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] \tag{84}$$

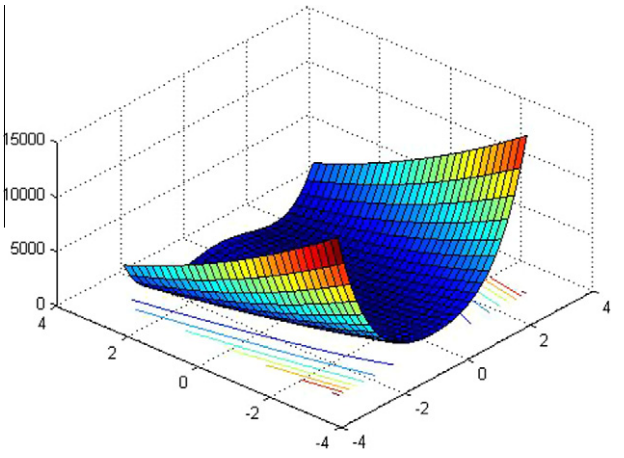

Fig. 11. The landscape of Rosenbrock's function: (a) the trajectory by using PSO-w; (b) the trajectory by using CPSO-inner (cubic); (c) the trajectory by using CPSO-inner (trigonal); (d) the trajectory by using CPSO-inner (hexagonal); (e) the trajectory by using CPSO-outer and (f) the magnified trajectory in the interval [1800,2600] by using CPSO-outer.

Since $\exists Q$, where Q is finite, so that $-Q < R^T < Q$. when $t > K$, $\max\{\|\lambda_1\|, \|\lambda_2\|\} = R < 1$.

$$\lim_{t\to\infty}(A\lambda_1^t + B\lambda_2^t) = \lim_{t\to\infty} R^t[(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] = \lim_{t\to\infty} R^T \cdot R^{t-T} \cdot [(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)]$$
$$= \lim_{t\to\infty} R^T \cdot 0 \cdot [(A+B)\cos(t\theta) + \imath(A-B)\sin(t\theta)] = 0 \quad (85)$$

Therefore, CPSO-outer will converge, if $\exists K$, where K is finite, so that, when $t > K$, $\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1$, in the case that $\omega(t)$ changes with iterations. □

3.2.3. Case study of trajectories

In order to study the search behavior of CPSO, a test function used in Section 4 was randomly chosen as an example to illustrate the behavior. The function is Rosenbrock's function, which is frequently used as a test function to test the performance of optimization algorithms. The global optimum lays inside a long, narrow, parabolic shaped flat valley, and it is very difficult to find global optimum. The landscape is plotted is Fig. 11. We draw the trajectory of one dimension of a randomly chosen particle in one execution of PSO-w [28], CPSO-inner and CPSO-outer, respectively. From Fig. 12, we can observe that PSO with inertia weight (PSO-w) could not let particles stabilize at a stable point for the problem, and the particle continuously oscillates till the end of the run. Particles in CPSO-inner (cubic, trigonal and hexagonal) could be adjusted by several damped waves and converge to a stable point. Fig. 12(e) presents some visible differences between PSO-w and CPSO-outer. In CPSO-outer, the particle oscillates intensively in a large amplitude in the early stage to conduct an explorative search due to the force that the cells impose on smart-cells, and then the neighborhood function performs as a tuner to fine-tune the particle's position. In order to observe the trajectory clearly, the interval in [1800,2500] is magnified in Fig. 12(f). The global optimum of Rosenbrock's function is reached when each dimension equals to 1, and from the trajectory shown in Fig. 12, only the dimension of the particle in CPSO-outer could successfully converge to this value. Fig. 13 shows that the range of coefficient $\forall i,j$ of CPSO-outer is small in the early stage, and then the range tends to increase along with iterations.

4. Experimental analysis and numerical results

In order to test the performance of CPSO and conduct a further comparative study, a set of well known test functions[20,40,9] are used as benchmark problems to test CPSO-inner (cubic, trigonal and hexagonal) and CPSO-outer, respectively. This set of functions consists of test functions with unfixed numbers of dimensions (10-dimensional and 30-dimensional in this paper) and test functions with fixed numbers of dimensions. Note that, for a problem with an unfixed number of dimensions, the dimension can be set at will. As for a problem with a fixed number of dimensions, the number of dimensions has been preset by the literatures and cannot be changed. Detailed descriptions of each function are given in the Appendix. These various test functions act as a collection of different landscapes with variable malicious cases, and would give an intensive test for optimization algorithms. The global optimal solution x^*, corresponding to the best fitness $f(x^*)$ in the search range of these test functions are listed in Table 1.

4.1. Algorithms used and parameters settings

In order to compare different variants of PSO and to test the effectiveness of CPSO, some variants of PSO algorithms generalized by [20] are tested on the set of test functions above. We present detailed computational data of all test functions for

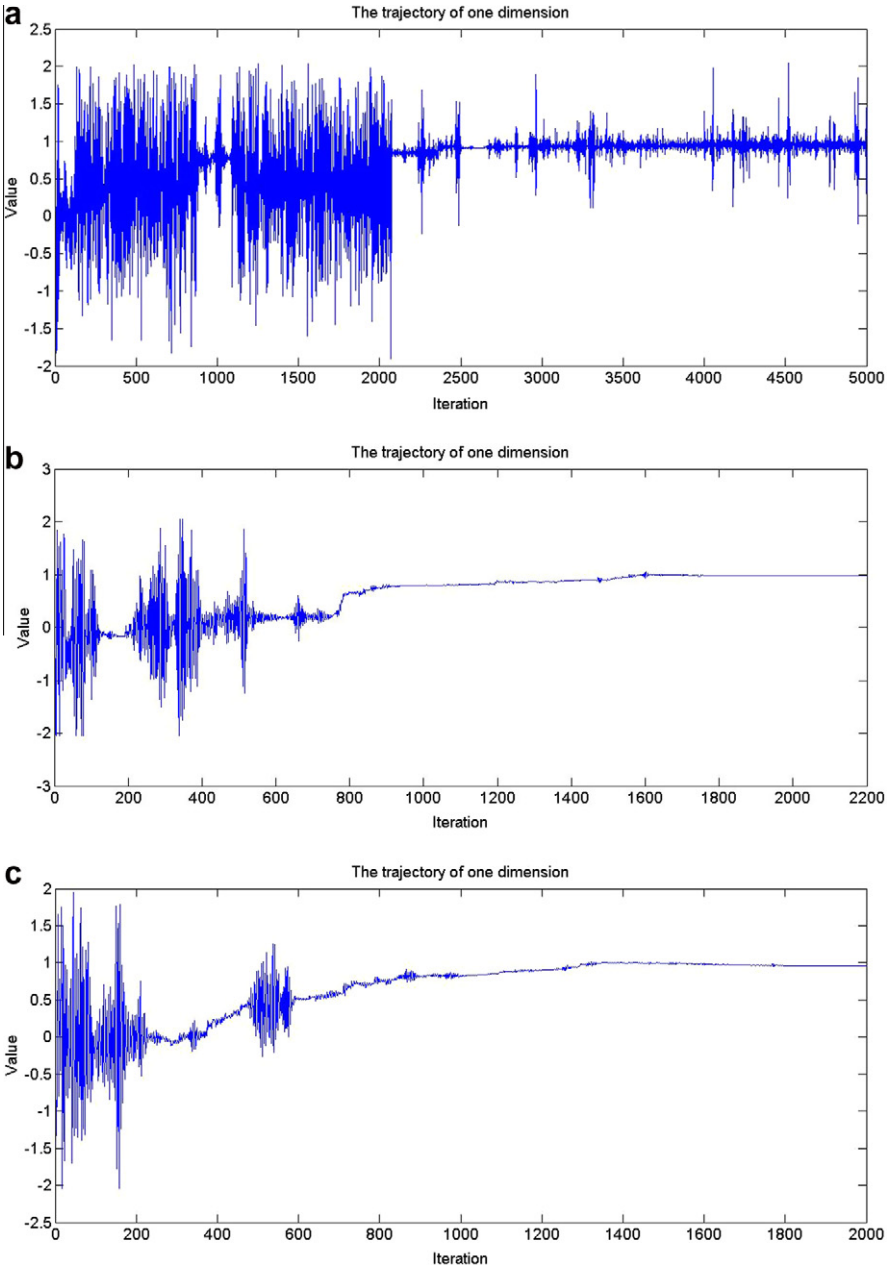

Fig. 12. The trajectory of one dimension of a particle.

all variants of PSO to conduct an exhaustive comparison. On one hand, we want to exhibit a comprehensive understanding for each variant of PSO; on the other hand, we provide an arena to examine the proposed CPSO. The successful rate of all variants of PSO, the best fitness, the worst fitness, mean value, standard deviation and the number of times of totally winning are given.

The 13 PSO algorithms are given as follows:

(1) PSO with inertia weight (PSO-w) [31];
(2) PSO with constriction factor (PSO-cf)[4];
(3) Local version of PSO with inertia weight (PSO-w-local);
(4) Local version of PSO with constriction factor (PSO-cf-local)[15];
(5) Unified PSO (UPSO)[26];
(6) Fully informed particle swarm (FIPS)[22];

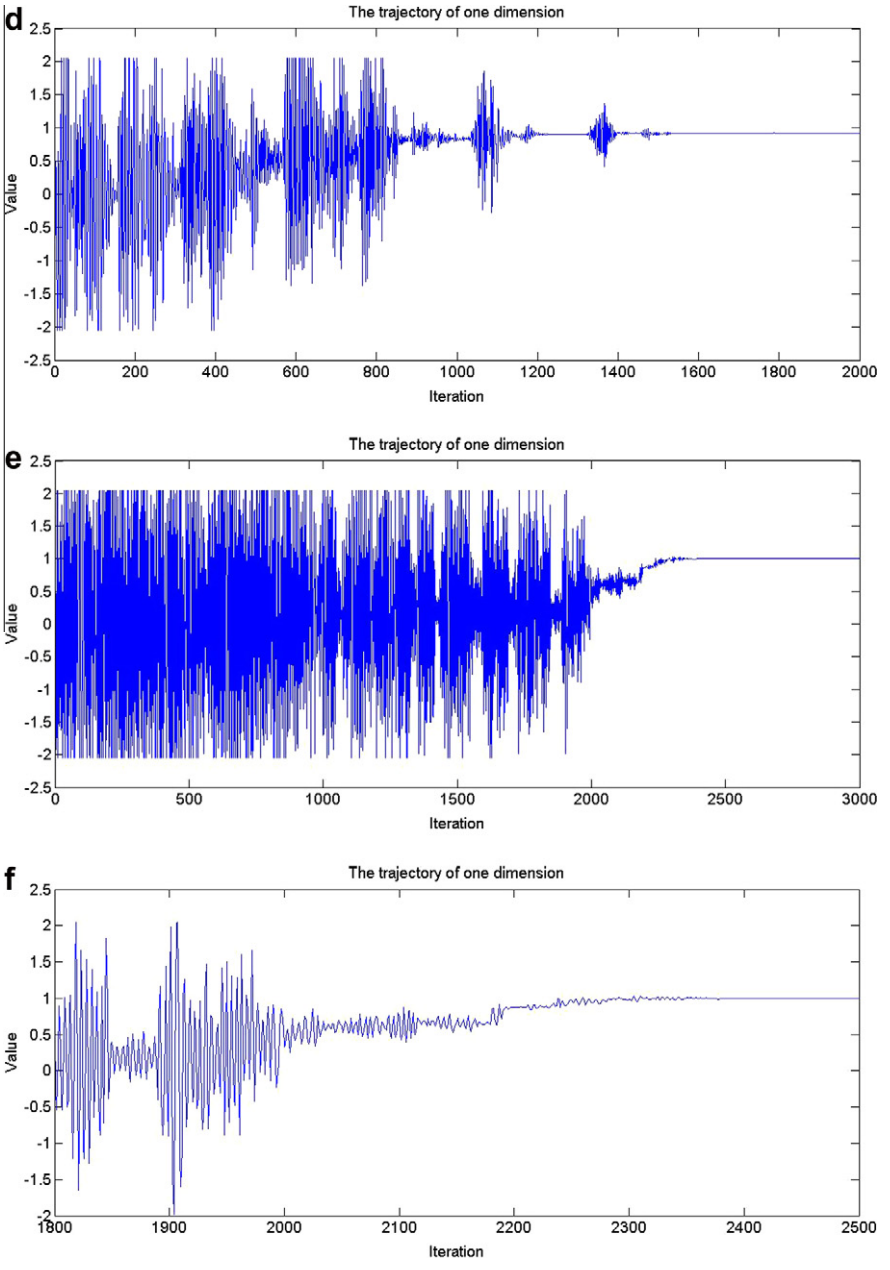

Fig. 12 (*continued*)

(7) Fitness–Distance-Ratio Based PSO (FDR-PSO)[27];
(8) Hybrid Cooperative Approach to PSO (CPSO-H)[35];
(9) Comprehensive Learning PSO (CLPSO)[20];
(10) Cellular PSO-inner (cubic);
(11) Cellular PSO-inner (trigonal);
(12) Cellular PSO-inner (hexagonal);
(13) Cellular PSO-outer (CPSO-outer).

The parameters of every variant of PSO are set in accordance with the original settings in the corresponding papers, except that the swarm size and max iteration times are set the same for all variants of PSO. The parameters settings in CPSO-inner and CPSO-outer are presented in Table 2. All experiments on each function were run 30 times.

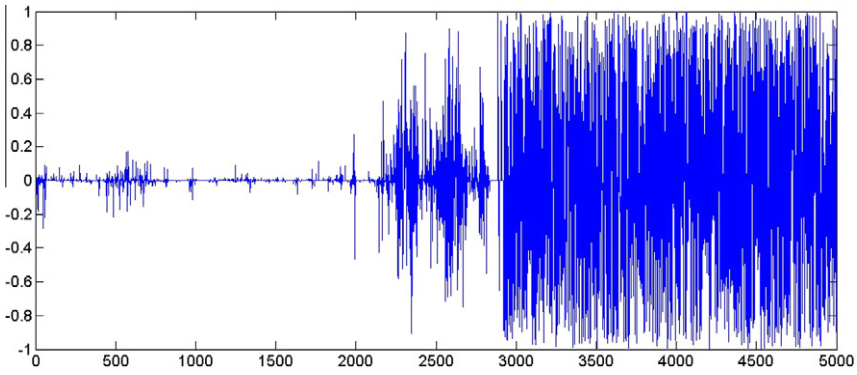

Fig. 13. The coefficient of CPSO-outer: (a) Convergence curve of sphere function; (b) convergence curve of Rosenbrock's function; (c) convergence curve of quadric's function; (d) convergence curve of Schwefel's function; (e) convergence curve of Griewank's function; (f) convergence curve of Weierstrass' function; (g) convergence curve of Quartic's function; (h) convergence curve of Noncontinuous Rastrigin's function; (i) convergence curve of Rastrigin's function and (j) convergence curve of Ackley's function.

Table 1
Detailed information of the test functions used in the paper.

No.	Function	Range	x^*	$f(x^*)$
1	$F_{Sphere}(x)$	$[-100,100]^D$	$[0,0,\ldots,0]$	0
2	$F_{Rosenbrock}(x)$	$[-2.048,2.048]^D$	$[1,1,\ldots,1]$	0
3	$F_{Quadric}(x)$	$[-100,100]^D$	$[0,0,\ldots,0]$	0
4	$F_{Schwefel}(x)$	$[-500,500]^D$	$[420.96,420.96,\ldots,420.96]$	0
5	$F_{Griewank}(x)$	$[-600,600]^D$	$[0,0,\ldots,0]$	0
6	$F_{Weierstrass}(x)$	$[-0.5,0.5]^D$	$[0,0,\ldots,0]$	0
7	$F_{Quartic}(x)$	$[-1.28,1.28]^D$	$[0,0,\ldots,0]$	0
8	$F_{Nonc-Rastri}(x)$	$[-5.12,5.12]^D$	$[0,0,\ldots,0]$	0
9	$F_{Rastrigin}(x)$	$[-5.12,5.12]^D$	$[0,0,\ldots,0]$	0
10	$F_{Ackley}(x)$	$[-32.768,32.768]^D$	$[0,0,\ldots,0]$	0
11	$F_{Beale}(x)$	$[-4.5,4.5]^2$	$[3,0.5]$	0
12	$F_{Bohachevsky1}(x)$	$[-100,100]^2$	$[0,0]$	0
13	$F_{Bohachevsky2}(x)$	$[-100,100]^2$	$[0,0]$	0
14	$F_{Bohachevsky3}(x)$	$[-100,100]^2$	$[0,0]$	0
15	$F_{Booth}(x)$	$[-10,10]^2$	[1,3]	0
16	$F_{Branin}(x)$	$[-5,10]$, $[0,15]$	$[-\pi,12.275]$, $[\pi,2.275]$, $[9.42478,2.475]$	0.397887
17	$F_{Colville}(x)$	$[-10,10]^4$	$[1,1,1,1]$	0
18	$F_{Easom}(x)$	$[-100,100]^2$	$[\pi,\pi]$	-1
19	$F_{G\&P}(x)$	$[-2,2]^2$	$[0,1]$	3
20	$F_{Hartmann1}(x)$	$[0,1]^3$	$[0.114614,0.555649,0.852547]$	-3.86278
21	$F_{Hartmann2}(x)$	$[0,1]^6$	$[0.20169,0.150011,0.476874,0.275332,0.311652,0.6573]$	-3.32237
22	$F_{Hump}(x)$	$[-5,5]^2$	$[0.0898,-0.7126]$, $[-0.0898,0.7126]$	-1.0316
23	$F_{Matyas}(x)$	$[-10,10]^2$	$[0,0]$	0
24	$F_{Michalewick1}(x)$	$[0,\pi]^2$	$[2.2029055,1.5707963]$	-1.8013
25	$F_{Michalewick2}(x)$	$[0,\pi]^5$	$[2.2029055,1.5707963,1.2849916,1.9230585,1.7204698]$	-4.6876582
26	$F_{Michalewick3}(x)$	$[0,\pi]^{10}$	$[2.2029055,1.5707963,1.2819916,1.9230585,1.7204698,1.5707963,$ $1.4544140,1.7560865,1.6557174,1.5707963]$	-9.6601517
27	$F_{NiH}(x)$	$[-5.12,5.12]^2$	$[0,0]$	-3600
28	$F_{Shekel1}(x)$	$[0,10]^5$	$[4,4,4,4]$	-10.1532
29	$F_{Shekel2}(x)$	$[0,10]^7$	$[4,4,4,4]$	-10.4029
30	$F_{Shekel3}(x)$	$[0,10]^{10}$	$[4,4,4,4]$	-10.4029

4.2. Computational complexity analysis

When using algorithms to optimize problems, the computational time is not always a reliable measure due to different runtime environment, programming language and coding style, which cause the computational time inconstant. So we turn to give computational complexity analysis which is an important criterion for evaluating the efficiency of algorithms. Critical program statements in an iteration are counted to evaluate the bound of the runtime in O notation. Further, computational complexity of all variants used in the paper is qualitatively ranked in Table 3. Note that l denotes the neighbors, and K denotes the number of fragments of a solution vector in CPSO-H [34].

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

Table 2
Parameters and their range of values used in our proposed algorithm.

Parameters	Value/range
Grid scale in CPSO-inner (i.e. swarm size N)	6×6
Smart-cells in CPSO-outer (i.e. swarm size N)	36
Acceleration constants c_1, c_2	$c_1 = c_2 = 1.49445$
Inertia weight w	[0.4, 1.2]
Maximum iteration times T	5000
Random number r_1, r_2	[0, 1]
The range of velocity V_{max}	$(X_{max} - X_{min})/2$
The boundary condition of particles	$X_i = \min(X_{max}, \max(X_{min}, X_i))$
The number of neighbors of CPSO-outer l	10

Table 3
The computational complexity of variants of PSO.

Algorithms	Procedure					Rank
	Initialize v and x	Evaluation	Choose p_i	Identify p_K	Loop (update)	
PSO-w	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
PSO-w-local	$O(ND)$	$O(ND)$	$O(ND)$	$O(NID)$	$O(ND + NID)$	$O(NID)$
PSO-cf	$O(ND)$	$O(ND)$	$O(ND)$	$O(N)$	$O(ND)$	$O(ND)$
PSO-cf-local	$O(ND)$	$O(ND)$	$O(ND)$	$O(NID)$	$O(ND + N1D)$	$O(NID)$
UPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND) + O(NID)$	$O(ND + NID)$	$O(N1D)$
FDR	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(N^2D)$	$O(N^2D)$
FIPS	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
CPSO-H	$O(ND) + O(ND)$	$O(NKD)+O(ND)$	$O(ND)+O(ND)$	$O(NKD) + O(ND)$	$O(NKD) + O(ND)$	$O(NKD)$
CLPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
CPSO-inner	$O(ND)$	$O(ND)$	$O(ND)$	$O(N1D)$	$O(ND + NID)$	$O(NID)$
CPSO-outer	$O(ND)$	$O(ND)$	$O(ND)$	$O(N)$	$O(NID)$	$O(NID)$

Table 4
The successful rate of all variants of PSO for 10-D problems.

10-D	Successful rate (%)												
Function	PSO-w	PSO-cf	PSO-w-l[a]	PSO-cf-l[a]	UPSO	FDR	FIPS	CPSO-H	CLPSO	Cubic[a]	Trigonal[a]	Hexagonal[a]	CPSO-outer
Sphere	0	0	0	0	0	0	0	0	0	36.7	40	0	100
Rosenbrock	0	0	0	0	0	0	0	0	0	0	0	0	86.7
Quadric	0	0	0	0	0	0	0	0	0	26.7	33.3	0	100
Schwefel	10	0	0	13.3	13.3	0	100	40	100	0	0	0	0
Griewank	0	0	0	0	0	0	0	0	20	33.3	30	0	100
Weierstrass	100	0	0	0	46.7	100	16.7	0	100	20	30	100	100
Quartic	0	0	0	0	0	0	0	0	0	0	0	0	0
Nonc.-Rastri.	10	0	0	0	13.3	13.3	0	100	100	23.3	30	100	100
Rastrigin	16.7	0	0	0	0	16.7	13.3	100	100	23.3	30	100	100
Ackley	0	0	0	0	20	0	23.3	0	0	30	33.3	0	100
Totally winning	1	0	0	0	0	1	1	2	4	0	0	3	7

[a] In this table, due to the limit of space, PSO-w-l, PSO-cf-l, cubic, trigonal and hexagonal represent PSO-w-local, PSO-cf-local, CPSO-inner (cubic), CPSO-inner (trigonal) and CPSO-inner (hexagonal), respectively.

From Table 3, we could observe that PSO has efficient computation, and the computational complexity is mainly affected by the swarm size (N) and the problem dimensionality (D). A larger swarm size or a larger problem dimensionality causes more computational efforts. The computational complexity of FDR is higher than other variants. This is because that FDR adds a new velocity component, which is used to select a particle among the swarm for updating each velocity dimension of each particle. PSO-w-local, PSO-cf-local, UPSO, CPSO-inner and CPSO-outer are summarized in the same rank. However, the parameter l and K could be fixed in advance. So, when the swarm size or problem dimensionality is enormously large, the contribution of l and K to the computational complexity can be neglected. Then the computational complexity of both of CPSO-outer and CPSO-inner will be $O(ND)$, so will be that of PSO-w-local, PSO-cf-local, UPSO, and CPSO-H. PSO-cf, FIPS and CLPSO have the same computational complexity as PSO-w. It is observed that the computational complexity of some of these variants of PSO could be summarized in the same rank, so we could conclude that different operations in PSO play a relatively less important role in computational complexity compared with swarm size and problem dimensionality, and all variants could solve problems in rational time.

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

Table 5
Results of Sphere function and Rosenbrocks function.

Function	Sphere				Rosenbrock			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(a) Results of Sphere function and Rosenbrock's function								
PSO-w	6.16E−140	3.98E−131	3.52E−132	9.46E−132	1.03E−02	3.99E+00	1.09E+00	8.18E−01
PSO-cf	4.79E−06	9.96E−06	8.36E−06	1.65E−06	2.05E−04	3.99E+00	2.68E−01	1.01E+00
PSO-w-local	4.24E−06	9.97E−06	8.42E−06	1.47E−06	1.27E+00	2.39E+00	1.70E+00	2.55E−01
PSO-cf-local	3.91E−06	9.94E−06	8.37E−06	1.56E−06	4.70E−03	4.05E−02	1.25E−02	7.41E−03
UPSO	8.98E−232	2.72E−225	1.97E−226	0	5.70E−04	4.37E−03	1.87E−03	9.53E−04
FDR	4.69E−255	1.80E−241	5.99E−243	0	5.41E−05	2.55E−02	1.10E−03	4.62E−03
FIPS	7.34E−55	7.28E−52	1.33E−52	1.81E−52	4.08E−01	6.34E−01	5.16E−01	5.64E−02
CPSO-H	4.55E−29	1.02E−24	1.17E−25	2.42E−25	3.80E−04	7.24E−02	3.60E−02	2.30E−02
CLPSO	3.58E−78	6.77E−75	3.98E−76	1.24E−75	1.16E−02	3.48E+00	1.40E+00	1.07E+00
CPSO-inner (cubic)	0(11)	1.06E−04	8.99E−06	2.48E−05	2.65E−19	6.36E+00	3.24E+00	1.83E+00
CPSO-inner (trigonal)	0(12)	8.03E−05	7.93E−06	2.29E−05	7.74E−01	8.30E+00	3.68E+00	2.39E+00
CPSO-inner (hexagonal)	5.11E−06	1.06E−02	1.26E−03	2.32E−03	4.90E+00	8.72E+00	6.89E+00	1.14E+00
CPSO-outer	**0**	**0**	**0**	**0**	0(26)	3.99E+00	2.66E−01	1.01E+00
(b) Results of Quadric function and Schwefel's function								
PSO-w	5.88E−136	2.12E−128	8.28E−130	3.86E−129	0(3)	4.74E+02	2.44E+02	1.26E+02
PSO-cf	5.14E−06	9.91E−06	7.81E−06	1.34E−06	4.74E+02	8.11E+02	6.92E+02	6.92E+02
PSO-w-local	6.93E−06	9.85E−06	8.96E−06	8.18E−07	1.18E+02	7.11E+02	5.08E+02	1.94E+02
PSO-cf-local	3.63E−06	9.74E−06	8.39E−06	1.51E−06	0(4)	3.55E+02	2.01E+02	1.48E+02
UPSO	1.36E−234	4.80E−225	1.61E−226	0	0(4)	5.92E+02	2.57E+02	1.94E+02
FDR	1.47E−267	1.71E−258	8.44E−260	0	2.37E+02	9.48E+02	5.24E+02	2.47E+02
FIPS	3.13E−53	4.85E−52	3.13E−53	8.84E−53	0	0	0	0
CPSO-H	1.57E−23	4.56E−19	2.87E−20	8.40E−20	0(12)	1.18E+02	7.11E+01	5.90E+01
CLPSO	1.22E−78	1.08E−74	4.58E−76	1.97E−75	0	0	0	0
CPSO-inner (cubic)	0(8)	1.12E−03	7.06E−05	2.27E−04	1.18E+02	1.55E+03	7.81E+02	2.92E+02
CPSO-inner (trigonal)	0(10)	4.61E−04	1.88E−05	8.46E−05	1.07E−01	1.31E+03	6.52E+02	3.09E+02
CPSO-inner (hexagonal)	5.56E−08	4.28E−02	1.21E−02	1.41E−02	9.09E+02	2.75E+03	1.94E+03	4.25E+02
CPSO-outer	**0**	**0**	**0**	**0**	1.18E+02	8.76E+02	5.21E+02	2.17E+02
(c) Results of Griewank's function and Weierstrass' function								
PSO-w	9.86E−03	1.50E−01	5.05E−02	2.61E−02	**0**	**0**	**0**	**0**
PSO-cf	2.46E−02	2.26E−01	8.37E−02	4.47E−02	2.33E−06	1.41E−01	9.89E−03	3.46E−02
PSO-w-local	7.40E−03	9.85E−02	5.36E−02	2.14E−02	3.20E−06	9.97E−06	8.94E−06	1.29E−06
PSO-cf-local	7.40E−03	6.89E−02	3.41E−02	1.65E−02	5.48E−06	9.97E−06	8.52E−06	1.13E−06
UPSO	9.86E−03	3.20E−02	2.02E−02	1.03E−02	0(14)	1.37E+00	1.06E−01	2.79E−01
FDR	7.40E−03	1.06E−01	5.53E−02	2.28E−02	**0**	**0**	**0**	**0**
FIPS	2.77E−10	2.37E−02	7.98E−03	7.86E−03	0(5)	4.10E−04	2.34E−05	8.15E−05
CPSO-H	9.86E−03	1.25E−01	3.95E−02	2.49E−02	2.13E−14	6.23E−09	3.98E−10	1.17E−09
CLPSO	0(6)	9.95E−11	1.53E−11	3.23E−11	**0**	**0**	**0**	**0**
CPSO-inner (cubic)	0(10)	3.22E−01	7.82E−02	8.91E−02	0(6)	1.45E+00	4.98E−01	4.53E−01
CPSO-inner (trigonal)	0(9)	2.07E−01	8.16E−02	7.31E−02	0(9)	1.18E+00	3.62E−01	4.01E−01
CPSO-inner (hexagonal)	9.91E−03	1.28E−01	6.83E−02	3.34E−02	**0**	**0**	**0**	**0**
CPSO-outer	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
(d) Results of Quartic function and noncontinuous Rastrigin's function								
PSO-w	1.51E−04	2.02E−03	7.77E−04	4.10E−04	0(3)	3.00E+00	3.67E−01	7.18E−01
PSO-cf	3.20E−04	2.88E−03	1.36E−03	6.15E−04	6.12E−06	9.00E+00	4.67E+00	2.09E+00
PSO-w-local	2.58E−04	2.46E−03	1.28E−03	5.97E−04	2.75E−07	3.00E+00	2.67E−01	6.91E−01
PSO-cf-local	1.27E−04	8.27E−04	3.66E−04	1.68E−04	7.72E−08	3.00E+00	2.00E−01	7.61E−01
UPSO	4.19E−04	3.39E−03	1.70E−03	8.36E−04	0(4)	7.00E+00	3.19E+00	1.67E+00
FDR	1.07E−04	1.95E−03	4.75E−04	3.79E−04	0(4)	2.00E+00	4.67E−01	6.81E−01
FIPS	1.36E−04	1.00E−03	5.31E−04	2.25E−04	4.65E−09	2.24E+00	6.52E−01	7.48E−01
CPSO-H	7.65E−04	6.02E−03	2.97E−03	1.40E−03	**0**	**0**	**0**	**0**
CLPSO	3.37E−04	1.54E−03	7.05E−04	2.61E−04	**0**	**0**	**0**	**0**
CPSO-inner (cubic)	1.26E−04	5.04E−03	9.66E−04	1.15E−03	0(7)	8.00E+00	2.80E+00	2.76E+00
CPSO-inner (trigonal)	8.86E−05	1.90E−03	5.72E−04	5.05E−04	0(9)	1.20E+01	2.87E+00	3.38E+00
CPSO-inner (hexagonal)	2.55E−05	4.64E−03	4.78E−04	8.35E−04	**0**	**0**	**0**	**0**
CPSO-outer	**7.35E−06**	**7.36E−04**	**1.99E−04**	**1.87E−04**	**0**	**0**	**0**	**0**
(e) Results of Rastrigin's function and Ackley's function								
PSO-w	0(5)	2.98E+00	1.45E+00	8.70E−01	2.66E−15	6.22E−15	5.39E−15	1.53E−15
PSO-cf	9.95E−01	1.29E+01	5.27E+00	2.88E+00	5.64E−06	9.92E−06	8.97E−06	9.59E−07
PSO-w-local	3.63E−06	1.99E+00	5.97E−01	8.10E−01	6.55E−06	9.94E−06	9.14E−06	8.59E−07
PSO-cf-local	9.95E−01	9.95E+00	3.45E+00	1.95E+00	7.74E−06	9.97E−06	9.21E−06	6.11E−07
UPSO	9.95E−01	8.95E+00	5.49E+00	2.46E+00	0(6)	2.66E−15	2.07E−15	1.35E−15
FDR	0(5)	3.98E+00	1.72E+00	1.11E+00	2.66E−15	6.22E−15	3.73E−15	1.66E−15
FIPS	0(4)	9.95E−01	1.01E−01	3.03E−01	0(7)	2.66E−15	2.31E−15	1.08E−15
CPSO-H	**0**	**0**	**0**	**0**	7.49E−13	6.47E−10	1.34E−10	1.87E−10

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

Table 5 (*continued*)

Function	Sphere				Rosenbrock			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
CLPSO	**0**	**0**	**0**	**0**	2.66E−15	2.66E−15	2.66E−15	0
CPSO-inner (cubic)	0(7)	1.29E+01	2.85E+00	3.79E+00	0(9)	2.32E+00	5.52E−01	7.82E−01
CPSO-inner (trigonal)	0(9)	1.19E+01	2.29E+00	3.13E+00	0(10)	2.81E+00	5.54E−01	8.62E−01
CPSO-inner (hexagonal)	0	0	0	0	2.74E−03	1.16E+00	1.22E−01	2.85E−01
CPSO-outer	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Note that different variants of PSO have different runtime due to different optimization strategies and different operations. CPSO-H takes more computational efforts because it hybridizes a swarm of PSO and several sub-swarms to optimize different fragments of a solution vector. CPSO-inner has the similar strategy as other variants with a local search (PSO-w-local and PSO-cf-local), so they have the similar runtime. Compared with PSO-w, CPSO-outer would require a little higher runtime because CPSO-outer would need to evaluate extra cells. However, from experimental results (see Section 4.3), we could see that CPSO-outer outperforms other variants on most of problems, and Fig. 13 shows it could perform well in fewer iterations. So we could say it is efficient. Moreover, since the rapid development of computer, the tradeoff between high-quality solutions and computational time tends to the former. So the quality of solutions preponderates when problems could be solved by algorithms in rational time.

4.3. Numerical results

We begin with a discussion on the results obtained for 10-D problems set. Table 4 gives the successful rate of all variants of PSO for 10-D problems, and Table 5 presents the computational results obtained by different variants of PSO. The numbers of times of reaching global optimum out of all runs are presented in the bracket listed in the "Best" column if the algorithm could get global optimum. For example, in Table 5, 0(11) means CPSO-inner (cubic) got global optimum 11 times during 30 runs on sphere function. It is used to show the capability of algorithms sufficiently, and provide reference about whether algorithms could locate global optimal point.

Sphere function is a continuous, convex, symmetrical and unimodal function, it is mainly used to test the accuracy of optimization algorithms. It is easily optimized by all variants of PSO, CPSO-outer could converge to the global optimum during all runs, and CPSO-inner (cubic and trigonal) also show good capability, they could get global optimum in some runs.

For Rosenbrock's function, FDR got the best mean value and showed good robustness, while CPSO-outer could get global optimum with a high successful rate and also perform well, however, it occasionally performs bad. Overall, it is also very competitive and effective for solving this function.

Table 5b shows good capability of CPSO-outer on Quadric function, it locates the global optimum during all runs, and CPSO-inner (cubic and trigonal) could find global optimum sometimes with relatively bad robustness. Schwefel's function is difficult for all variants except CPSO-H and CLPSO, most of the variants of PSO converge with low optimization accuracy.

CPSO-outer shows better search performance on Griewank's function and Weierstrass' function, it could obtain the global optimum during all runs; CPSO-inner (cubic and trigonal) could reach global optimum on both functions sometimes, while CPSO-inner (hexagonal) could reach global optimum during all runs on Weierstrass' function.

Quartic function is unimodal with random noise. Noisy functions are widespread in real-world problems, and every evaluation for the function is disturbed by noise, so the particles' information is inherited and diffused noisily, which makes the problem hard for optimization. CPSO-outer outperforms other algorithms, and CPSO-inner shows a moderate performance.

For noncontinuous Rastrigin's function, more than half variants of PSO have the possibility for locating the global optimum, while CPSO-outer, CPSO-inner (hexagonal), CLPSO and CPSO-H are robust to obtain global optimum during all runs.

Rastrigin's function, based on Sphere function, uses cosine function to generate lots of local optimal points. It is a complex multimodal function, and optimization falls into the local optimum easily. CPSO-outer, CPSO-inner (hexagonal), CLPSO and CPSO-H could find global optimum successfully, moreover, CPSO-inner (cubic and trigonal) also has the possibility for finding global optimum.

Better results were also obtained by CPSO-outer for Ackley's function, which is a continuous, rotate and non-separable multimodal function. The exterior region of the function is nearly flat, while the center is a high peak, it has many widespread local optimal points from the flat region to the center peak. CPSO-outer successfully achieved the global optimum during all runs, while CPSO-inner fails to get good performances.

In order to give a visualized and detailed comparison of the proposed CPSO, Fig. 14 gives the convergence curves of CPSO and the algorithms with performance ranking top three of the other variants of PSO on every problem. The plot depicts convergence trends of variants of PSO in a random run. From Fig. 14, we could observe that CPSO-outer converges to global optimum quickly on Sphere function, Quadric's function, Griewank's function, Weierstrass' function, Noncontinuous Rastrigin's function, Rastrigin's function and Ackley' function in no more than 500 iterations, so it is effective for both unimodal and multimodal function, while CPSO-inner shows a moderate performance in the comparison.

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

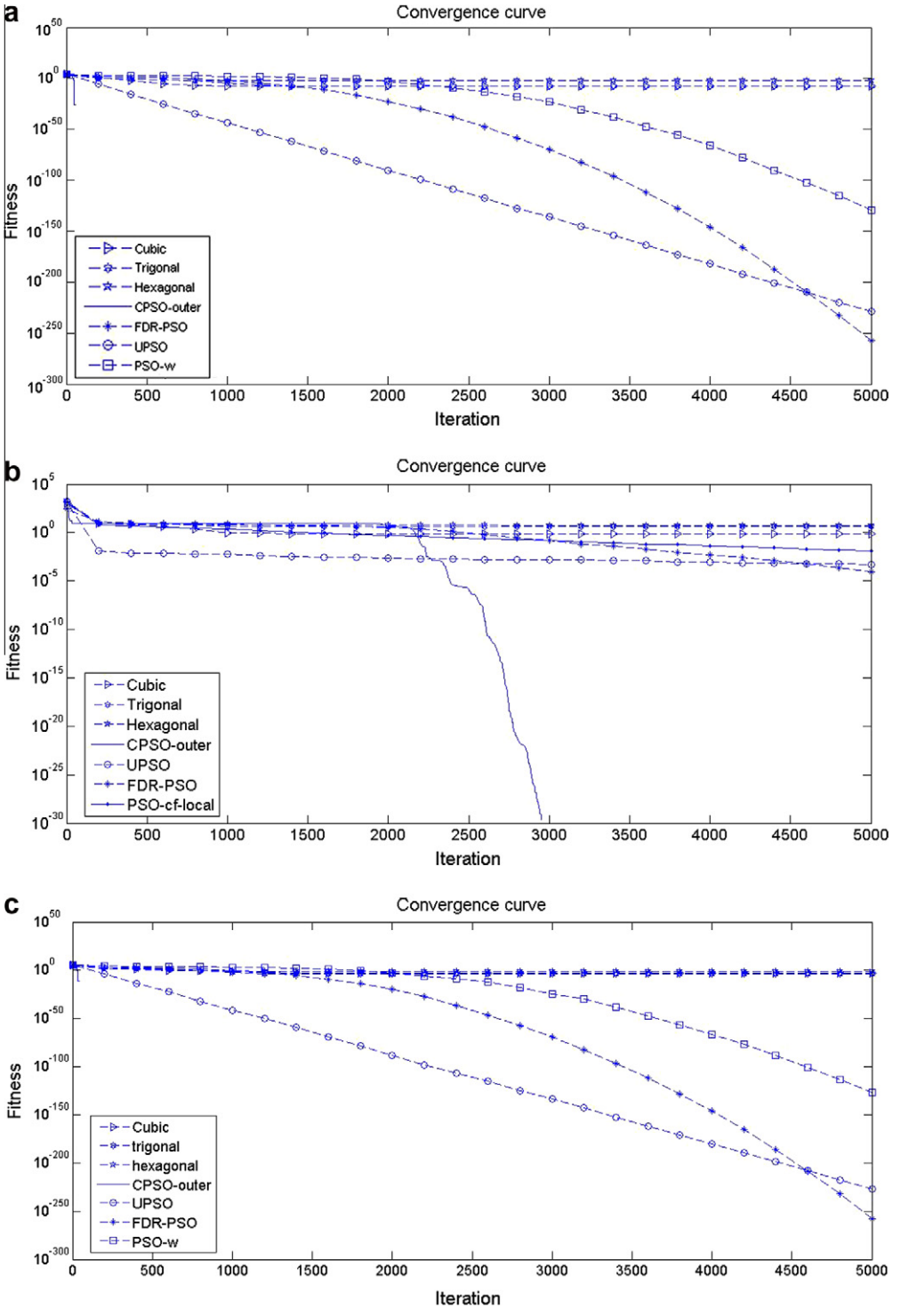

Fig. 14. Convergence curves for 10-D problems.

And then, we consider computational results of 30-D problems. For most of the optimization algorithms, their performance will sharply decrease when the search space is enlarged and dimensionality increases. So the results would not be as good as in 10-D problems. Table 6 shows that the successful rates of all variants of PSO for 30-D problems are very low. From Table 7, it is observed that CPSO-outer outperforms other variants of PSO on Rosenbrock's function, Quartic

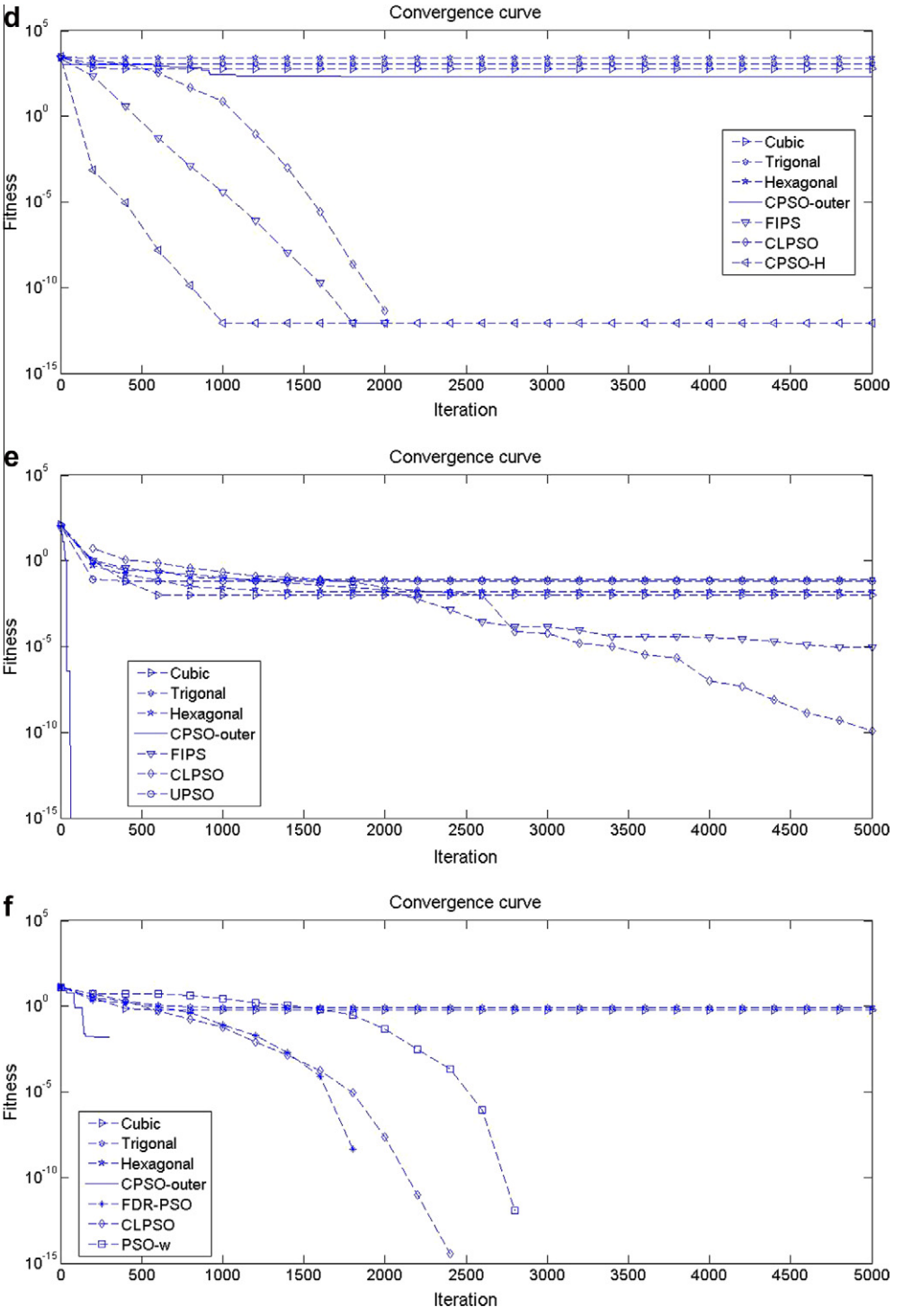

Fig. 14 (*continued*)

function and Ackley's function. However, for Sphere function, CPSO-outer is observed to be have high possibility for locating the global optimum and the mean value ranks the third place in all variants of PSO; the same condition happened to Quadric function. For Griewank's function, it ranks the second place following CLPSO. CPSO-inner performs less robust and gets bad results on Sphere function, Rosenbrock's function, Quadric function, Griewank's function and Ackley's function. So the performance of CPSO-inner heavily depends on its neighborhood topologies and the landscape of test functions. Different

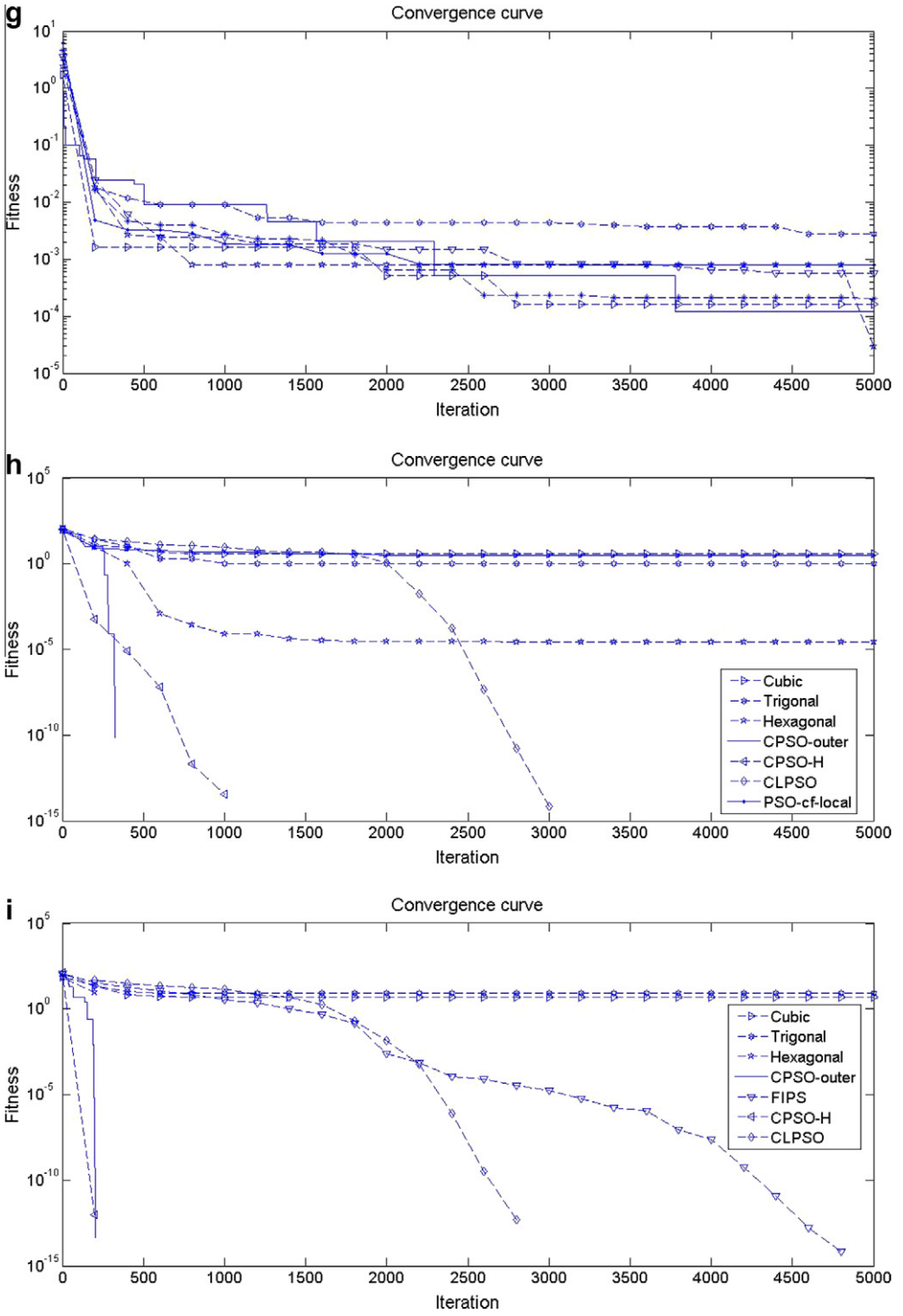

Fig. 14 (*continued*)

topologies are apt to different landscapes, and dimensionality also affects the performance heavily. Due to limitations of space, we didn't show the detailed convergence curves for 30-D problems.

In the end, we analyze computational results of problems with fixed numbers of dimensions. Table 8 shows the successful rate of all variants of PSO for problems with fixed numbers of dimensions, and the detailed results of four test functions which are very difficult for most of the variants of PSO are given in Table 9. From Table 8, it is observed that every variant

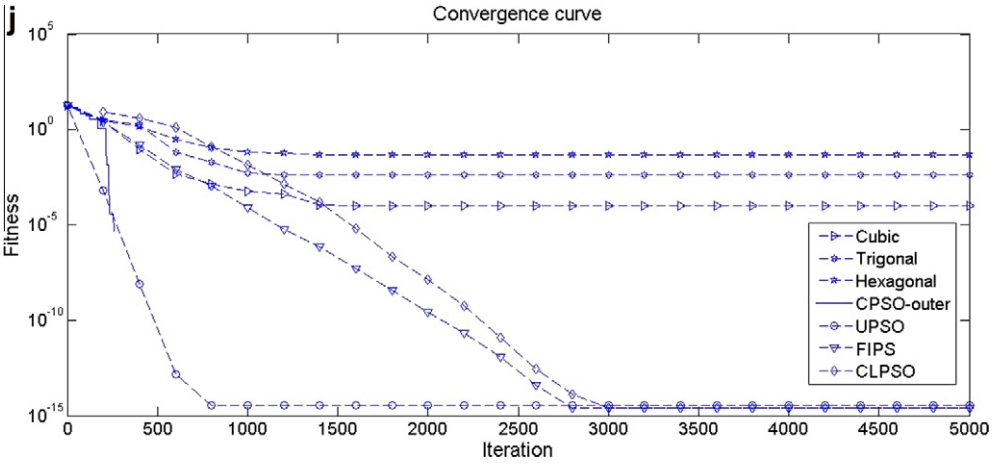

Fig. 14 (*continued*)

Table 6
The successful rate of all variants of PSO for 30-D problems.

30-D	Successful rate (%)												
Function	PSO-w	PSO-cf	PSO-w-l[a]	PSO-cf-l[a]	UPSO	FDR	FIPS	CPSO-H	CLPSO	Cubic[a]	Trigonal[a]	Hexagonal[a]	CPSO-outer
Sphere	0	0	0	0	0	0	0	0	0	20	10	0	90
Rosenbrock	0	0	0	0	0	0	0	0	0	0	0	0	0
Quadric	0	0	0	0	0	0	0	0	0	0	0	0	40
Schwefel	0	0	0	0	0	0	0	16.7	90	0	0	0	0
Griewank	0	0	0	0	0	0	0	0	10	20	16.7	0	26.7
Weierstrass	0	0	0	0	0	0	0	0	100	0	0	40	0
Quartic	0	0	0	0	0	0	0	0	0	0	0	0	0
Nonc-Rastri	0	0	0	0	0	0	0	0	0	0	0	70	0
Rastrigin	0	0	0	0	0	0	0	0	0	6.7	60	0	10
Ackley	0	0	0	0	0	0	0	0	0	16.7	0	0	20
Totally winning	0	0	0	0	0	0	0	0	1	0	0	0	0

[a] In this table, due to the limit of space, PSO-w-l, PSO-cf-l, cubic, trigonal and hexagonal represent PSO-w-local, PSO-cf-local, CPSO-inner (cubic), CPSO-inner (trigonal) and CPSO-inner (hexagonal), respectively.

of PSO could perform well on most of these test functions, which means that the mechanism of velocity update and position update of PSO is effective. The landscapes of Beale function, Bohachevsky1/2/3 function, Booth function and Branin function extend smoothly, so it is easy for all variants of PSO to optimize them. Colville function shows difficulty for most of the variants of PSO, while it could be well solved by CPSO-inner and CPSO-outer. Easom is a unimodal function; G&P is a simple global optimization test function; and Hump function has six local minima, two of which are global ones. Observed from Table 8, all variants of PSO can converge to the global optimum with 100% successful rate. Along with the increase of dimensionality, optimization difficulty is highly enhanced for the Hartmann1/2 function and Michalewics1/2/3 function. As to Needle-in-a-haystack function (NiH), the global optimum is surrounded by the worst solutions, which makes algorithms hardly search this area. The global optimum locates at (0,0) with the value of -3600, and there are four local optimum distributed at $(-5.12,-5.12)$, $(-5.12,5.12)$, $(5.12,-5.12)$ and $(5.12,5.12)$ with the value of -2748.78. From Table 9, it is observed that only PSO-w-local and CPSO-outer could successfully locate at the global optimum during all run. However, PSO-cf failed to optimize this function because particles would fly out of the search space seriously by the end of a run. Most of the algorithms easily converge to the local optimum at early stage, and hardly escape in the following search. Then we consider Shekel1/2/3 function, the results show that FIPS, CLPSO, CPSO-inner and CPSO-outer could converge to the global optimum during all runs, while other variants of PSO could only succeed certain probability. In sum, CPSO-outer could win on most of this set of functions – it could get a totally winning for 17 functions of the 20 functions.

In summary, the experimental results illustrate that CPSO-outer performs very well on 10-D problems since it successfully finds global optimum of seven out of ten functions and outperforms other variants of PSO on eight out of ten functions, and CPSO-outer is also accurate and efficient in minimization of 30-D test functions and functions with fixed numbers of dimensions. The success of CPSO-outer could come from a better balance of global exploration and local exploitation, resulting from a powerful CA-based mechanism that guides particles. However, CPSO-inner has been shown to perform well in

Table 7
Experimental results for 30-D problems.

Function	Sphere				Rosenbrock			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(a) Results of Sphere function and Rosenbrock's function								
PSO-w	1.13E−33	3.45E−28	1.72E−29	6.53E−29	1.05E+00	4.38E+00	2.07E+00	1.20E+00
PSO-cf	7.37E−06	9.93E−06	9.26E−06	6.63E−07	9.07E+00	1.46E+01	1.14E+01	1.95E+00
PSO-w-local	6.42E−06	9.99E−06	9.40E−06	8.25E−07	2.41E+01	2.94E+01	2.60E+01	2.26E+00
PSO-cf-local	7.38E−06	9.94E−06	9.03E−06	6.56E−07	1.77E+01	2.32E+01	1.95E+01	2.44E+00
UPSO	1.82E−89	5.00E−85	3.22E−86	9.09E−86	1.40E+01	1.66E+01	1.55E+01	8.96E−01
FDR	1.41E−107	1.42E−95	**5.84E−97**	**2.63E−96**	5.36E+00	8.98E+00	6.76E+00	1.34E+00
FIPS	1.59E−12	2.16E−11	4.46E−12	3.66E−12	2.38E+01	2.48E+01	2.44E+01	3.10E−01
CPSO-H	4.20E−09	1.84E−07	3.98E−08	3.94E−08	1.37E−02	7.26E+01	1.72E+01	2.16E+01
CLPSO	3.27E−23	1.57E−21	3.70E−22	3.31E−22	1.22E+01	2.22E+01	1.83E+01	3.59E+00
CPSO-inner (cubic)	0(6)	7.82E+02	1.83E+02	2.13E+02	2.87E+01	8.25E+01	3.55E+01	1.25E+01
CPSO-inner (trigonal)	0(3)	5.96E+02	1.37E+02	1.52E+02	2.86E+01	7.34E+01	3.96E+01	1.19E+01
CPSO-inner (hexagonal)	5.38E−01	3.16E+00	1.60E+00	6.94E−01	2.86E+01	3.08E+01	2.90E+01	6.57E−01
CPSO-outer	0(27)	2.81E−69	9.48E−71	5.13E−70	**1.05E−04**	**3.46E+00**	**1.01E+00**	**6.65E−01**
	Quadric				Schwefel			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(b) Results of Quadric function and Schwefel's function								
PSO-w	3.16E−30	3.58E−27	8.00E−28	1.42E−27	7.11E+02	1.90E+03	1.33E+03	3.76E+02
PSO-cf	7.33E−06	9.96E−06	8.99E−06	1.00E−06	4.17E+03	5.74E+03	5.03E+03	5.20E+02
PSO-w-local	8.66E−06	9.95E−06	9.43E−06	6.06E−07	4.46E+03	6.24E+03	5.61E+03	6.35E+02
PSO-cf-local	8.06E−06	9.98E−06	9.20E−06	8.75E−07	9.48E+02	3.43E+03	2.43E+03	8.65E+02
UPSO	3.27E−86	7.61E−84	1.91E−84	2.82E−84	2.57E+03	3.81E+03	3.40E+03	3.47E+02
FDR	1.53E−103	1.62E−92	**2.70E−93**	**6.60E−93**	2.76E+03	3.63E+03	3.27E+03	3.65E+02
FIPS	1.71E−10	8.54E−10	6.02E−10	2.55E−10	1.22E+02	7.03E+02	3.34E+02	2.11E+02
CPSO-H	1.27E−06	1.02E−05	5.41E−06	3.71E−06	0(5)	5.92E+02	2.50E+02	2.00E+02
CLPSO	4.65E−21	9.03E−20	3.41E−20	2.31E−20	0(27)	1.18E+02	1.18E+01	3.60E+01
CPSO-inner (cubic)	2.86E+01	7.78E+01	4.31E+01	1.70E+01	2.11E+03	6.93E+03	4.48E+03	9.58E+02
CPSO-inner (trigonal)	2.87E+01	1.18E+02	4.43E+01	2.08E+01	2.23E+03	5.70E+03	3.83E+03	9.27E+02
CPSO-inner (hexagonal)	2.78E+01	3.07E+01	2.88E+01	7.58E−01	4.79E+03	1.02E+04	7.60E+03	1.49E+03
CPSO-outer	0(12)	3.67E−51	8.10E−52	1.24E−51	4.74E+02	6.24E+03	1.51E+03	1.42E+03
	Griewank				Weierstrass			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(c) Results of Griewank's function and Weierstrass' function								
PSO-w	1.72E−02	1.35E−01	7.06E−02	3.13E−02	7.11E−15	2.25E−03	9.53E−04	1.02E−03
PSO-cf	1.72E−02	1.18E−01	6.24E−02	2.39E−02	4.21E−06	1.65E−02	6.94E−04	3.07E−03
PSO-w-local	8.47E−06	1.06E−01	3.79E−02	2.43E−02	4.74E−06	1.00E−05	8.60E−06	1.30E−06
PSO-cf-local	9.02E−06	8.61E−02	3.35E−02	2.32E−02	6.26E−06	9.98E−06	8.80E−06	8.06E−07
UPSO	6.40E−04	6.23E−02	2.49E−02	1.61E−02	7.77E+00	1.26E+01	9.06E+00	1.85E+00
FDR	7.40E−03	1.38E−01	7.26E−02	3.36E−02	7.11E−15	1.50E+00	3.20E−01	6.63E−01
FIPS	2.55E−06	4.94E−02	1.56E−02	1.38E−02	1.25E−04	1.96E−04	1.62E−04	3.00E−05
CPSO-H	9.86E−03	1.16E−01	5.71E−02	2.64E−02	7.11E−15	3.96E−09	2.48E−10	7.66E−10
CLPSO	**0(3)**	**8.00E−10**	**6.65E−11**	**1.91E−10**	**0**	**0**	**0**	**0**
CPSO-inner (cubic)	0(6)	9.24E+00	2.84E+00	3.15E+00	1.09E+00	1.51E+01	9.62E+00	4.67E+00
CPSO-inner (trigonal)	0(5)	1.84E+01	2.76E+00	4.24E+00	3.08E−01	1.37E+01	6.28E+00	4.50E+00
CPSO-inner (hexagonal)	5.15E−02	2.66E−01	1.50E−01	5.87E−02	0(12)	5.06E+00	1.15E+00	1.85E+00
CPSO-outer	0(8)	1.17E−01	1.52E−02	2.22E−02	1.32E−01	3.00E+00	1.91E+00	1.38E+00
	Quartic				Non-Rastri			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(d) Results of Quartic function and noncontinuous Rastrigin's function								
PSO-w	1.37E−02	4.41E−02	2.68E−02	1.18E−02	7.00E+00	3.90E+01	1.70E+01	1.18E+01
PSO-cf	2.09E−03	9.71E−03	6.27E−03	2.62E−03	2.30E+01	7.10E+01	4.15E+01	1.92E+01
PSO-w-local	1.79E−02	4.02E−02	2.97E−02	7.66E−03	1.30E+01	2.40E+01	1.87E+01	5.20E+00
PSO-cf-local	4.00E−03	6.89E−03	5.09E−03	1.17E−03	4.00E+00	1.90E+01	9.33E+00	5.68E+00
UPSO	3.36E−03	1.59E−02	1.01E−02	4.99E−03	6.50E+01	1.00E+02	7.96E+01	1.59E+01
FDR	1.70E−03	5.37E−03	3.28E−03	1.07E−03	7.00E+00	1.70E+01	1.08E+01	4.40E+00
FIPS	4.02E−03	5.74E−03	4.79E−03	6.35E−04	5.62E+01	7.61E+01	6.23E+01	7.21E+00
CPSO-H	9.68E−03	2.26E−02	1.44E−02	4.84E−03	3.33E−10	2.04E−08	5.39E−09	7.49E−09
CLPSO	2.51E−03	8.45E−03	5.44E−03	1.91E−03	**4.80E−13**	**7.12E−12**	**2.57E−12**	**2.61E−12**
CPSO-inner (cubic)	8.49E−03	1.48E−01	4.00E−02	4.39E−02	3.01E+00	8.90E+01	4.91E+01	2.56E+01
CPSO-inner (trigonal)	9.85E−05	2.95E−02	8.64E−03	1.04E−02	2.15E+00	3.96E+01	1.97E+01	1.30E+01
CPSO-inner (hexagonal)	5.04E−05	7.13E−03	7.67E−04	1.65E−03	0(21)	1.01E+00	1.42E−01	3.46E−01
CPSO-outer	**6.05E−05**	**4.45E−04**	**1.54E−04**	**1.10E−04**	2.50E+01	1.25E+02	7.52E+01	3.52E+01
	Rastrigin				Ackley			

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

Table 7 (*continued*)

Function	Sphere				Rosenbrock			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(e) Results of Rastrigin's function and Ackley's function								
PSO-w	1.29E+01	3.68E+01	2.89E+01	7.45E+00	2.04E−14	5.24E−14	4.00E−14	1.25E−14
PSO-cf	4.68E+01	9.35E+01	5.95E+01	1.70E+01	9.33E−06	2.01E+00	6.52E−01	1.01E+00
PSO-w-local	2.09E+01	3.38E+01	2.64E+01	4.08E+00	8.74E−06	9.95E−06	9.63E−06	4.01E−07
PSO-cf-local	2.89E+01	6.17E+01	4.03E+01	1.20E+01	9.12E−06	1.00E−05	9.61E−06	3.95E−07
UPSO	5.17E+01	9.95E+01	6.63E+01	1.45E+01	2.66E−15	2.66E−15	2.66E−15	0.00E+00
FDR	2.09E+01	3.38E+01	2.77E+01	4.30E+00	2.04E−14	3.82E−14	2.66E−14	7.04E−15
FIPS	5.66E+01	8.51E+01	7.30E+01	9.70E+00	3.38E−07	5.50E−07	4.39E−07	8.25E−08
CPSO-H	2.07E−09	7.12E−08	1.46E−08	1.92E−08	2.27E−05	9.68E−05	5.74E−05	3.06E−05
CLPSO	1.07E−14	7.66E−13	**2.12E−13**	**2.18E−13**	1.63E−12	5.85E−12	3.61E−12	1.29E−12
CPSO-inner (cubic)	0(2)	8.68E+01	3.82E+01	2.43E+01	0(5)	7.99E+00	3.60E+00	2.49E+00
CPSO-inner (trigonal)	6.66E−02	7.90E+01	3.02E+01	2.08E+01	9.41E−01	7.57E+00	3.81E+00	1.71E+00
CPSO-inner (hexagonal)	0(18)	2.47E+01	1.38E+00	4.75E+00	1.77E+00	3.84E+00	3.13E+00	6.79E−01
CPSO-outer	0(3)	1.08E+02	5.00E+01	2.49E+01	**0(6)**	**6.22E−15**	**5.03E−15**	**2.90E−15**

Table 8
The successful rate of all variants of PSO for problems with fixed-dimension.

Function	Successful rate (%)												
	PSO-w	PSO-cf	PSO-w-l[a]	PSO-cf-l[a]	UPSO	FDR	FIPS	CPSO-H	CLPSO	Cubic[a]	Trigonal[a]	Hexagonal[a]	CPSO-outer
Beale	100	100	100	100	100	80	100	100	100	100	100	100	100
Bohachevsky1	100	100	100	100	100	100	100	100	100	100	100	100	100
Bohachevsky2	100	100	100	100	100	100	100	100	100	100	100	100	100
Bohachevsky3	100	100	100	100	100	100	100	100	100	100	100	100	100
Booth	100	100	100	100	100	100	100	100	100	100	100	100	100
Branin	100	100	100	100	100	100	100	100	100	100	100	100	100
Colville	0	0	0	0	0	0	0	0	0	100	100	100	100
Easom	100	100	100	100	100	100	100	100	100	100	100	100	100
G&P	100	100	100	100	100	100	100	100	100	100	100	100	100
Hartmann1	100	100	100	100	100	100	100	100	100	100	100	100	100
Hartmann2	63.3	56.67	70	90	100	50	93.33	40	100	40	86.67	53.3	46.67
Hump	100	100	100	100	100	100	100	100	100	100	100	100	100
Matyas	0	100	63.33	100	100	100	87.5	100	0	100	100	100	100
Michalewics1	100	100	100	100	100	100	100	100	100	100	100	100	100
Michalewics2	90	25	90	63.33	100	33.33	90	100	100	13.33	30	26.67	83.33
Michalewics3	6.67	0	0	0	0	0	13.33	100	100	0	0	0	0
NiH	6.67	–	100	60	33.3	40	26.67	20	0	13.33	10	6.67	100
Shekel1	33.33	36.67	96.67	73.33	96.67	56.67	100	33.33	100	100	100	100	100
Shekel2	76.67	100	100	96.67	90	100	100	23.33	100	100	100	100	100
Shekel3	93.33	56.67	100	100	100	80	100	20	100	100	100	100	100
Totally winning	11	13	13	13	15	12	14	14	17	16	16	16	17

[a] In this table, due to the limit of space, PSO-w-l, PSO-cf-l, cubic, trigonal and hexagonal represent PSO-w-local, PSO-cf-local, CPSO-inner (cubic), CPSO-inner (trigonal) and CPSO-inner (hexagonal), respectively.

some of the test functions, but is ineffective and less robust for the others. The cause for this inconsistence in its performance is probably the inconformity between neighborhood topologies and landscape of test functions.

5. Conclusion

This paper proposes a novel variant of PSO called cellular particle swarm optimization, and explores how particle swarm works in the view of cellular automata. Two different cellular ideas lead to two versions of CPSO, namely, CPSO-inner and CPSO-outer. In CPSO-inner, particles update by using the information inside the swarm. The performance of CPSO-inner depends on the specific functions, as different topologies are apt to different landscapes. Also the algorithm's accuracy and efficiency are greatly affected by the dimensionality of the problems. When the dimensions increase, the optimization capability decreases rapidly. Differing from CPSO-inner, CPSO-outer exploits the information outside the swarm during the updating process – smart-cell communicates with cells outside the swarm wisely. A large set of experimental data shows the superior of CPSO-outer in performance on most of the test functions. This smart-cell strategy gives particles more powerful external force to let themselves move wisely, and jump out of the local optimum. The performance of CPSO-outer could lead to a conclusion that it is a significant idea that particles turning their "attention" to the outside of the swarm.

Please cite this article in press as: Y. Shi et al., Cellular particle swarm optimization, Inform. Sci. (2010), doi:10.1016/j.ins.2010.05.025

Table 9
Q2 Experimental results for problems with fixed-dimension.

Function	Colville				Hartmann2			
Criteria	Best	Worst	Mean	Std	Best	Worst	Mean	Std
(a) Results of Colville function and Hartmann2 function								
PSO-w	4.81E−10	1.46E−05	2.37E−06	3.61E−06	−3.32237	−3.20316	−3.27866	0.05843
PSO-cf	7.43E−21	2.42E−14	4.05E−15	5.92E−15	−3.32237	−3.20316	−3.27071	0.06008
PSO-w-local	3.68E−07	7.87E−04	1.73E−04	2.28E−04	−3.32237	−3.13768	−3.28442	0.06006
PSO-cf-local	1.51E−12	6.29E−09	1.37E−09	1.62E−09	−3.32237	−3.20316	−3.31045	0.03637
UPSO	4.38E−13	2.36E−08	2.21E−09	5.55E−09	**−3.32237**	**−3.32237**	**−3.32237**	**6.39E−016**
FDR	7.45E−21	2.91E−15	2.63E−16	6.71E−16	−3.32237	−3.20316	−3.26276	0.06062
FIPS	5.06E−07	2.67E−04	6.61E−05	7.84E−05	−3.32237	−3.32176	−3.32235	0.00011
CPSO-H	1.18E−22	1.96E−19	4.77E−20	4.88E−20	−3.32237	−3.20316	−3.25084	0.0594
CLPSO	6.17E−05	4.26E−02	4.42E−03	9.74E−03	−3.32237	−3.13271	−3.24152	0.07053
CPSO-inner (cubic)	**0**	**0**	**0**	**0**	−3.32237	−3.20316	−3.30647	0.04122
CPSO-inner (trigonal)	**0**	**0**	**0**	**0**	−3.32237	−3.19740	−3.26245	0.06095
CPSO-inner (hexagonal)	**0**	**0**	**0**	**0**	−3.32237	−3.20316	−3.26674	0.06049
CPSO-outer	**0**	**0**	**0**	**0**	−3.32237	−3.20316	−3.25879	0.06049
(b) Results of Michalewics function3 and Needle-in-a-haystack function								
PSO-w	−9.6601517	−9.0339692	−9.4492138	0.1656032	−3600	−2729.1	−2798.1	218.0172
PSO-cf	−9.3877554	−8.0364848	−8.9527162	0.3368332	–	–	–	–
PSO-w-local	−9.6225401	−9.1780898	−9.5126648	0.1149541	**−3600**	**−3600**	**−3600**	**0**
PSO-cf-local	−9.6183889	−8.8332284	−9.2455775	0.2350785	−3600	−2734.9	−3269.8	417.5431
UPSO	−9.4725380	−8.4814091	−9.006959	0.2550431	−3600	−2732	−3105.3	412.3334
FDR	−9.6134777	−8.5923698	−9.2371609	0.275055	−3600	−2748.8	−3089.3	424.1387
FIPS	−9.6601517	−9.4676076	−9.6165626	0.0451111	−3600	−2748.8	−2975.8	382.8577
CPSO-H	**−9.6601517**	**−9.6601517**	**−9.6601517**	**0**	−3600	−2748.8	−2919	346.3078
CLPSO	**−9.6601517**	**−9.6601517**	**−9.6601517**	**0**	−3598.9	−2748.8	−2.9681	355.7935
CPSO-inner (cubic)	−9.5318198	−7.8688812	−8.7272344	0.5410021	−3600	−2748.8	−2862.3	294.305
CPSO-inner (trigonal)	−9.5879108	−7.5138153	−8.6496907	0.6443260	−3600	−2748.8	−2833.9	259.7308
CPSO-inner (hexagonal)	−9.5128920	−6.9740738	−8.5506601	0.5968011	−3600	−2748.8	−2.8055	215.9608
CPSO-outer	−9.6552406	−8.5538554	−9.4751090	0.2186244	**−3600**	**−3600**	**−3600**	**0**

In the future, inspired by CA and the two versions of CPSO, we could wisely define different cell states (S_i) and transition rules to design potentially effective variants of PSO under the general framework of CPSO. Moreover, it will also be interesting to investigate the following issues: studying novel CA strategies to exploit information from each dimension of a particle; designing effective information interaction mechanism to strengthen the power of PSO; discussing approaches to enable CPSO to handle discrete variables to solve real-world optimization problems effectively.

Acknowledgement

This paper is supported by Program for New Century Excellent Talents in University under Grant No. NCET-08-0232. The authors would like to express our sincere gratitude to Prof. P.N. Suganthan for his help and codes of different variants of PSO, and to anonymous reviewers for their constructive and valuable comments.

Appendix A. Functions with unfixed numbers of dimensions

(1) Sphere function

$$F_{Sphere}(x) = \sum_{i=1}^{D} x_i^2$$

(2) Rosenbrock's function

$$F_{Rosenbrock}(x) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$$

(3) Quadric function

$$F_{Quadric}(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$$

(4) Schwefel's function

$$F(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin\left(|x_i|^{\frac{1}{2}}\right)$$

(5) Griewank's function

$$F_{Griewank}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

(6) Weierstrass' function

$$F_{Weierstrass}(x) = \sum_{i=1}^{D-1}\left(\sum_{k=0}^{k\max}\left[a^k \cos(2\pi b^k (x_i + 0.5))\right]\right) - D\sum_{k=0}^{k\max}\left[a^k \cos(2\pi b^k \cdot 0.5)\right]$$

$$a = 0.5, \quad b = 3, \quad k\max = 20$$

(7) Quartic function i.e. noise

$$F_{Quartic}(x) = \sum_{i=1}^{D} i x_i^4 - random[0,1)$$

(8) Noncontinuous Rastrigin's function

$$F_{Nonc\text{-}Rastri}(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos(2\pi y_i) + 10\right)$$

$$y_i = \begin{cases} x_i & |x_i| < \frac{1}{2} \\ \frac{round(2x_i)}{2} & |x_i| \geqslant \frac{1}{2} \end{cases} \quad \text{for } i = 1, 2, \ldots, D$$

(9) Rastrigin's function

$$F_{Rastrigin}(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

(10) Ackley's function

$$F_{Ackley}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e$$

Appendix B. Functions with fixed numbers of dimensions

(11) Beale function

$$F_{Beale}(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

(12) Bohachevsky1 function

$$F_{Bohachevsky1}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

(13) Bohachevsky2 function

$$F_{Bohachevsky2}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$$

(14) Bohachevsky3 function

$$F_{Bohachevsky3}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$$

(15) Booth function

$$F_{Booth}(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

(16) Branin function

$$F_{Branin}(x) = \left(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

(17) Colville function

$$F_{Colville}(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$$

(18) Easom function

$$F_{Easom}(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$$

(19) Goldstein and Price function

$$F_{G\&P}(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) * (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2))$$

(20) Hartmann1 function

$$F_{Hartmann1}(x) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right]$$

$$\alpha = [1,\ 1.2,\ 3,\ 3.2]^T,\ A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},\ P = 10^{-4}\begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$$

(21) Hartmann2 function

$$F_{Hartmann2}(x) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{6} B_{ij}(x_j - Q_{ij})^2\right]$$

$$\alpha = [1,\ 1.2,\ 3,\ 3.2]^T,\quad B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 107 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$Q = 10^{-4}\begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

(22) Hump function

$$F_{Hump}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

(23) Matyas function

$$F_{Matyas}(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$$

(24) Michalewics1 function

$$F_{Michalewics1}(x) = -\sum_{j=1}^{2} \sin(x_j)\left(\sin\left(\frac{jx_j^2}{\pi}\right)\right)^{20}$$

(25) Michalewics2 function

$$F_{Michalewics2}(x) = -\sum_{j=1}^{5} \sin(x_j)\left(\sin\left(\frac{jx_j^2}{\pi}\right)\right)^{20}$$

(26) Michalewics3 function

$$F_{Michalewics3}(x) = -\sum_{j=1}^{10} \sin(x_j)\left(\sin\left(\frac{jx_j^2}{\pi}\right)\right)^{20}$$

(27) Needle-in-a-haystack

$$F_{NiH}(x) = -\left(\left(\frac{a}{b+(x^2+y^2)}\right)^2 + (x^2+y^2)^2\right)$$

$a = 3.0, \quad b = 0.05$

(28) Shekel1 function

$$F_{Shekel1}(x) = -\sum_{j=1}^{5}\left[\sum_{i=1}^{4}(x_i - C_{ij})^2 + \beta_j\right]^{-1}$$

(29) Shekel2 function

$$F_{Shekel1}(x) = -\sum_{j=1}^{7}\left[\sum_{i=1}^{4}(x_i - C_{ij})^2 + \beta_j\right]^{-1}$$

(30) Shekel3 function

$$F_{Shekel1}(x) = -\sum_{j=1}^{10}\left[\sum_{i=1}^{4}(x_i - C_{ij})^2 + \beta_j\right]^{-1}$$

$$\beta = \frac{1}{10}[1,\ 2,\ 2,\ 4,\ 4,\ 6,\ 3,\ 7,\ 5,\ 5]^T,$$

$$C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \end{bmatrix} \quad \text{for Shekel } 1/2/3 \text{ function}$$

References

[1] M.A. Abido, Optimal design of power-system stabilizers using particle swarm optimization, IEEE Transactions on Energy Conversion 17 (3) (2002) 406–413.
[2] P. Angeline, Using selection to improve particle swarm optimization, In: Proceedings of IEEE International Conference on Evolutionary Computation, 1998, pp. 84–89.
[3] J.-F. Chang, P. Shi, Using investment satisfaction capability index based particle swarm optimization to construct a stock portfolio, Information Sciences (2010), doi:10.1016/j.ins.2010.05.008.
[4] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.
[5] W.L. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Sciences 178 (2008) 3096–3109.
[6] M. El-Abd, M.S. Kamel, A cooperative particle swarm optimizer with migration of heterogeneous probabilistic models, Swarm Intelligence 4 (1) (2010) 57–89.
[7] S.-K.S. Fan, E. Zahara, A hybrid simplex search and particle swarm optimization for unconstrained optimization, European Journal of Operational Research 181 (2007) 527–548.
[8] Z.-L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on Power Systems 18 (3) (2003) 1187–1195.
[9] A. Header, M. Fukushima, Tabu search directed by direct search methods for nonlinear global optimization, European Journal of Operational Research 170 (2006) 329–349.
[10] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, In: Proceedings of IEEE Swarm Intelligence Symposium, 2003, pp. 72–79.
[11] R. Kathiravan, R. Ganguli, Strength design of composite beam using gradient and particle swarm optimization, Composite Structures 81 (2007) 471–479.
[12] J. Kennedy, R.C. Eberhart, Particle swarm optimization, In: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
[13] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, In: Proceedings of IEEE Conference on Systems, Man, and Cybernetics, 1997, pp. 4104-4109.
[14] J. Kenndy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, In: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1931–1938.
[15] J. Kennedy, R. Mendes, Population structure and particle swarm performance, In: Proceedings of IEEE Congress on Evolutionary Computation, 2002, pp. 1671–1676.
[16] M. Kutrib, Cellular automata—a computational point of view, Studies in Computational Intelligence 113 (2008) 183–227.
[17] A. Lazinica, Particle Swarm Optimization, IN-TECH, 2009.
[18] X.D. Li, Niching without niching parameters: particle swarm optimization using a ring topology, IEEE Transactions on Evolutionary Computation 14 (1) (2010) 150–169.
[19] Z.G. Lian, B. Jiao, X.S. Gu, A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan, Applied Mathematics and Computation 182 (2) (2006) 1008–1017.
[20] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281–295.
[21] M. Løvbjerg, T.K. Rasmussen, T. Krink, Hybird particle swarm optimiser with breeding and subpopulations, In: Proceedings of the GECCO, 2001, pp. 469–476.
[22] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation 8 (2004) 204–210.

[23] S. Nakano, A. Ishigame, K. Yasuda, Particle swarm optimization based on the concept of Tabu search, In: Proceedings of IEEE Congress on Evolutionary Computation, 2007, pp. 3258–3263.
[24] E. Ozcan, C.K. Mohan, Analysis of a simple particle swarm optimization system, Intelligent Engineering Systems through Artificial Neural Networks 8 (1998) 253–258.
[25] E. Ozcan, C.K. Mohan, Particle swarm optimization: surfing the waves, In: Proceedings of IEEE Congress on Evolutionary Computation, 1999, pp. 1939–1944.
[26] K.E. Parsopoulos, M.N. Vrahatis, UPSO – a unified particle swarm optimization scheme, Lecture Series on Computational Sciences (2004) 868–873.
[27] T. Peram, K. Veeramachaneni, C.K. Mohan, Fitness–distance-ratio based particle swarm optimization, In: Proceedings of Swarm Intelligence Symposium, 2003, pp. 174–181.
[28] T.K. Rasmussen, T. Krink, Improved Hidden markov model training for multiple sequence alignment by a particle swarm optimization-evolutionary algorithm hybrid, Biosystems 72 (2003) 289–301.
[29] J.L. Schiff, Cellular Automata: A Discrete View of the World, Wiley & Sons, 2007.
[30] D.Y. Sha, C.Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem, Computers and Industrial Engineering 51 (4) (2006) 791–808.
[31] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, In: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska,1998, pp. 66–73.
[32] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, In: Proceedings of the IEEE Congress on Evolutionary Computation, 1999, pp. 1945–1950.
[33] P.N. Suganthan, Particle swarm optimiser with neighborhood operator, In: Proceedings of IEEE Congress on Evolutionary Computation, 2002, pp. 1958–1961.
[34] P.K. Tripathi, S. Bandyopadhyay, S.K. Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Information Sciences 177 (2007) 5033–5049.
[35] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.
[36] F. Van den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (2006) 937–971.
[37] V. Vassiliadis, G. Dounias, Nature-inspired intelligence: a review of selected methods and applications, International Journal on Artificial Intelligence Tools 18 (4) (2009) 487–516.
[38] Y.J. Wang, Y.P. Yang, Particle swarm optimization with preference order ranking for multi-objective optimization, Information Sciences 179 (2009) 1944–1959.
[39] S. Wolfram, A New Kind of Science, Wolfram Media, Inc., 2002.
[40] X. Yao, Y. Liu, G.M. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (1) (1999) 82–102.
[41] P.Y. Yin, F. Glover, M. Laguna, J.X. Zhu, Cyber swarm algorithms – improving particle swarm optimization using adaptive memory strategies, European Journal of Operational Research 201 (2) (2010) 377–389.