Self-Organizing Potential Field Network: A New Optimization Algorithm

Lu Xu and Tommy Wai Shing Chow, *Senior Member, IEEE*

***Abstract*—This paper presents a novel optimization algorithm called self-organizing potential field network (SOPFN). The SOPFN algorithm is derived from the idea of the vector potential field. In the proposed network, the neuron with the best weight is considered as the target with the attractive force, while the neuron with the worst weight is considered as the obstacle with the repulsive force. The competitive and cooperative behaviors of SOPFN provide a remarkable ability to escape from the local optimum. Simulations were performed, compared, and analyzed on eight benchmark functions. The results presented illustrate that the SOPFN algorithm achieves a significant performance improvement on multimodal problems compared with other evolutionary optimization algorithms.**

***Index Terms*—Neural network, self-organizing map, stochastic optimization, vector potential field.**

I. Introduction

OPTIMIZATION refers to the study of problems in which one seeks to minimize or maximize a function by efficiently choosing the values of floating-point or integer variables within a space limit. The function can be defined according to different problems, e.g., complex networks, telecommunications, traveling salesman problems, and path planning of mobile robots. In practice, it is impossible to find the optimal solution by sampling every value in a search space within a finite computational time.

The stochastic optimization technique, which has become increasingly important in solving optimization problems, includes local strategies such as simulated annealing (SA) [13], and population-based strategies such as genetic algorithm (GA), differential evolution (DE), particle swarm optimization (PSO), and self-organizing migrating algorithm (SOMA). These algorithms are all designed to emulate different physical or biological behaviors to provide optimization characteristics. Traditional GA [15], [19], [29] works iteratively by applying three operators—selection, crossover, and mutation—for each individual. This characteristic of GA is prone to yielding the premature convergence and reducing the convergence rate. The basic DE [8], [12], [16] generates new individuals by adding the distance between two randomly selected individuals to a third one. The DE method self-adjusts as the perturbation gradually decreases from the distance between the individuals in the population. Although the mechanism of DE increases the diversity, the convergence rate is slow because of its over-stochastic characteristic. PSO [5], [11], [26], which emulates the swarm behavior in a collaborative manner, lets every particle go toward the direction of the best position of all particles, as well as the direction of its own best position found so far. PSO performs effectively on most optimization problems except multimodal problems. SOMA [25], [34], [35] updates every individual by a "migration loop" to generate a series of candidate solutions. The migration strategy makes SOMA require less training iteration.

Besides the above algorithms, there are other population-based optimization algorithms such as evolution strategies [3], memetic algorithm [7], ant colony system [4], and tabu search [10]. Compared with local strategies, population-based strategies replace the single search with the population search and generate more opportunities to find the optimal solution. And because of the availability of parallel computation, population-based strategies are efficient at solving complex optimization problems.

Recently, some evolutionary algorithms (EAs) combined with the mechanism of a neural network have been developed. Self-organizing and self-evolving neurons (SOSEN) [31] embed the self-organizing behavior [2], [14], [28] in SA to improve the convergence rate. In SOSENs, each neuron evolves using SA and cooperates with other neurons by a self-organizing operator. Since the search space is enlarged by multiple neurons, the performance of SOSENs has proven to be better than that of SA. A new evolution strategy with neighborhood attraction (EN), neuron gas attraction EN (NG-EN) [9], which uses a neural gas approach to dynamically adapt the neighborhood structure, defines that the neighborhood radius of each individual is determined by its average Euclidean distance to all other individuals. This method enhances the convergence rate to find the optimum. The neural gas self-organizing net (NG-ES) [23], a modification of the NG learning algorithm, generates an additional adaptation term to avoid premature convergence.

In [1], Bergh *et al.* discuss the issue of "two steps forward, one step back" in most optimization algorithms. This means that some components of an individual vector may move close to the optimum, while others may move away from the optimum when the vector learns from the best candidate solution. This mechanism makes the probability of generating

Manuscript received August 2, 2008; revised February 1, 2010, March 17, 2010, and March 19, 2010; accepted March 22, 2010. Date of publication June 21, 2010; date of current version September 1, 2010.

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: xulu22@student.cityu.edu.hk; eetchow@cityu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2047264

an individual inside the optimal region exponentially decrease as the dimensionality increases. Another issue for most optimization algorithms is that individuals in the population only learn from the best candidate solution even though this solution is far from the global optimum. This mechanism makes individuals more susceptible to being trapped in the local optimum, especially when the search space is multimodal. To solve these problems, Potter and De Jong [24] suggest that every vector in GA can be split into several parts forming multiple populations instead of a single population. Bergh *et al.* propose the cooperative PSO (CPSO) algorithm [1], which partitions an n-dimensional individual vector into n swarms of 1-D vector. In the CPSO algorithm, a high-dimensional problem is transformed into a 1-D problem. It significantly improves the diversity, but increases the number of function evaluations. The comprehensive learning PSO (CLPSO) [17], [18] is introduced by Liang *et al.* to improve the performance of PSO. In CLPSO, the best positions of all particles are used to update the velocity of one particle. In such a way, CLPSO explores a larger search space, but at the expense of convergence rate.

In this paper, a new optimization algorithm called the self-organizing potential field network (SOPFN) is developed. The SOPFN is an evolutionary algorithm that models the search space as a self-organizing potential field similar to the vector potential field (VPF) [22]. The VPF proposed by Masoud and Bayoumi is designed for constructing navigation controls. VPF prevents a robot from entering an undesired region and guarantees convergence to a target. The VPF algorithm consists of two parts. The first part controls the robot to drive the motion toward the destination in an obstacle-free space. The second part deflects the motion away from the obstacles. In the SOPFN algorithm, the neurons with the best and worst candidate solutions are considered as the target with the attractive force and the obstacle with the repulsive force, respectively. The self-organizing interactions of neurons in the search space are motivated by potential functions, which control neurons to move toward the target and to avoid the obstacle. Compared with other evolutionary optimization algorithms, the operator of repulsive force provides SOPFN with better ability to escape from the local optimum, and the SOPFN algorithm exhibits significant performance improvement in terms of accuracy in multimodal optimization problems.

Section II presents an overview of self-organizing map (SOM) and some traditional optimization algorithms. Section III describes the principle, architecture, and implementation of the SOPFN. Section IV shows the simulation results of seven algorithms on eight benchmark functions. Lastly, the algorithm analysis of SOPFN and conclusion are given in Sections V and VI, respectively.

II. Background

A. SOM

The SOM [2], [14], [28], [32], [33] proposed by Kohonen is an unsupervised learning neural network with the ability to visualize high-dimensional data in a low-dimensional map. The map consists of a number of neurons, each of which has a feature vector with the same dimensionality as the input data. By assigning each input datum to a neuron with the closest feature vector, SOM is able to display the topology of input data. As a result, data with close proximity are mapped together due to the neighborhood neurons.

Denote input data $x \in \Re^D$, and neurons with weight vectors $w = [w_1, w_2, \ldots, w_D]^T$. At each training step, a sample datum x is randomly chosen from the input dataset. Find the winning neuron c whose vector is the closest to x in a 2-D map ($M \times N$) according to

$$c = \arg\min_i \|x - w_i\|, \quad i \in \{1, 2, \ldots, M \times N\}. \tag{1}$$

The updating formula for every neuron is

$$w_i = \begin{cases} -2pcw_i + \alpha \cdot \eta_{ic} \cdot [x - w_i], & \text{if } i \in N_C \\ w_i, & \text{otherwise} \end{cases} \tag{2}$$

where α is the learning rate that monotonically decreases with time, N_C and η_{ic} are the neighborhood set and the neighborhood function of winner neuron c, and

$$\eta_{ic} = \exp\left(-\frac{\|P_i - P_c\|^2}{2\sigma^2}\right) \tag{3}$$

where P_i and P_c are coordinates of neuron i and neuron c in the map, σ is the neighborhood radius.

B. SOMA

SOMA [25], [34], [35] is a population-based evolutionary algorithm. SOMA generates a series of new candidate solutions for the optimization function from an individual at each iteration.

The ith individual vector x_i is expressed as $x_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]^T$ in a population. SOMA assigns perturbations to some components of x_i by a PRT parameter

$$v_{ij} = \begin{cases} 1, & \text{if } rand_j \le PRT \\ 0, & \text{if } rand_j > PRT \end{cases} \quad j = 1, 2, \ldots, D \tag{4}$$

where $rand_j \in [0, 1]$ is the random number of the jth component, and $PRT \in [0, 1]$ is the given constant. Then, use the following rule:

$$x_i^{'} = x_i + v_i \cdot t \cdot (x_G - x_i) \tag{5}$$

where x_G is the best solution and $t \in \{\Delta, 2\Delta, \ldots, k\Delta\}$. Δ is the step size and k is the step number. A number of $x_i^{'}$ are produced over different values of t, and the best solution is chosen as the new x_i.

C. PSO

PSO [1], [5], [11], [17], [18], [26] proposed by Kennedy and Eberhart is derived by emulating the behaviors of flocks of birds, schools of fish, and herds of animals, which live in a collaborative manner. At each time step, every particle moves toward the direction of the best position among all particles' previous positions, as well as the direction of its own best position found so far. Each particle has two features, position

and velocity. The position of ith particle is denoted as x_i with D dimensions in a swarm, and v_i is the velocity of ith particle $v_i = [v_{i1}, v_{i2}, \ldots, v_{iD}]^T$. The updating rule is formalized by

$$\begin{aligned} v_{ij} &= \omega \cdot v_{ij} + c_1 \cdot rand_j^1 \cdot (x_{gj} - x_{ij}) \\ &\quad + c_2 \cdot rand_j^2 \cdot (x_{pj} - x_{ij}), \quad j=1, 2, \ldots, D \end{aligned} \tag{6}$$

$$x_{ij} = x_{ij} + v_{ij}, \quad j = 1, 2, \ldots, D \tag{7}$$

where x_g is the best position yielded so far among all particles' historical positions, x_p is the best position found so far by particle i, ω is the inertia weight, c_1 and c_2 are positive constants, $rand_j^1$ and $rand_j^2 \in [0, 1]$ are random numbers.

D. SOSEN

The SOSEN [31] is a neural network algorithm running SA for each neuron. SOSENs can be considered as a combination of self-evolving behavior and self-organizing behavior, which are similar to the crossover and mutation operators of GA.

In the case of SOSENs, the weight of a neuron is updated by SOM after running SA. Results obtained by SOSENs can be better than those obtained by running a single SA because the input space is largely searched using multiple neurons.

At each training step, SOSENs finds the winning neuron c with the best solution, and updates all neurons by

$$w_i = \begin{cases} w_c, & \text{if } i = c \\ w_i + \alpha \cdot \eta_{ic} \cdot (w_c - w_i), & \text{otherwise} \end{cases} \tag{8}$$

where α is the learning rate, and η_{ic} is the neighborhood function.

The distance between a neuron and the winning neuron determines the magnitude of the neuron being updated toward the winner in the solution space. That is, if the neuron is the one nearest to the winner, the magnitude of updating is the largest.

III. Self-Organizing Potential Field Network

A. Self-Organizing Potential Field Strategy

The proposed SOPFN algorithm uses a similar network structure of SOM [2], [14], [28]. In SOM, the learning rule is used to adjust the weights of neurons for a given dataset. As a result, neurons represent a data topology. In SOPFN, the learning rule is developed to tune the weights of neurons toward the optimum solution of the given optimization function. Each weight of neuron is considered as a candidate solution.

The network of SOPFN consists of an array of neurons located at 2-D rectangular grids. The neuron whose weight is the best solution is the winner among all neurons. Fig. 1 shows an example of the SOPFN network with 5×5 neurons.

Each neuron in the SOPFN network has two learning mechanisms, cooperation and competition. The cooperation behavior is a self-organizing procedure that the neurons subjected to the winning neuron's neighborhood are trained. The competition behavior models the network as a potential field similar to the vector potential field used in mobile robots [20], [21], [27]. The neuron with the best solution is considered as

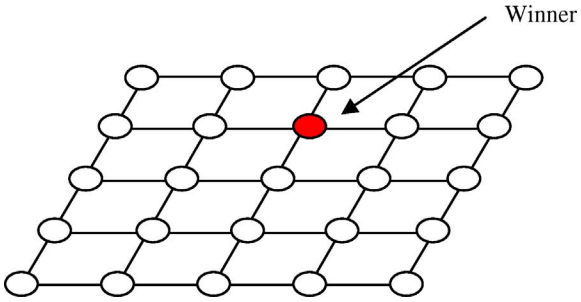

Fig. 1. Network structure of SOPFN with 5×5 neurons.

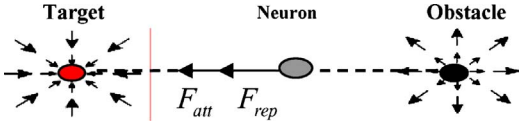

Fig. 2. Potential field with attractive and repulsive forces.

the target attracting other neurons, while the neuron with the worst solution is considered as the obstacle repelling other neurons. The potential forces of neurons are shown in Fig. 2. Every neuron is operated by two forces, attractive force F_{att} and repulsive force F_{rep}. The potential field organizes neurons to move toward the target by the attractive force and away from the obstacle by the repulsive force.

The SOPFN network can be expressed by a $M \times N$ neuron map with the weight $w \in \Re^D$, $w = [w_1, w_2, \ldots, w_D]^T$. The potential forces acting on the ith neuron are formalized by the following rules:

$$F_{\text{att}}^i = \alpha_1 \cdot \eta_1 \cdot [w_c - w_i] \quad i \in \{1, 2, \ldots, M \times N\} \tag{9}$$

where w_c is the weight of target neuron c, α_1 is the attractive learning rate, η_1 is the attractive neighborhood function, and

$$\eta_1 = \exp\left(\frac{\|P_i - P_c\|^2}{2\sigma^2}\right) \tag{10}$$

where P_i and P_c are coordinates of neuron i and neuron c respectively in the map, σ is the neighborhood radius. And

$$F_{\text{rep}}^i = \alpha_2 \cdot \eta_2 \cdot [w_r - w_i] \quad i \in \{1, 2, \ldots, M \times N\} \tag{11}$$

where w_r is the weight of obstacle neuron r, α_2 is the repulsive learning rate, η_2 is the repulsive neighborhood function, and

$$\eta_2 = \exp\left(-\frac{\|P_i - P_r\|^2}{2\sigma^2}\right) \tag{12}$$

where P_i and P_r are coordinates of neuron i and neuron r respectively in the map.

The Euclidean distance between the neuron and the target or obstacle neuron has a significant effect on potential forces. In (10), the attractive force increases as the distance increases. On the contrary, (12) shows that the repulsive force decreases as the distance increases.

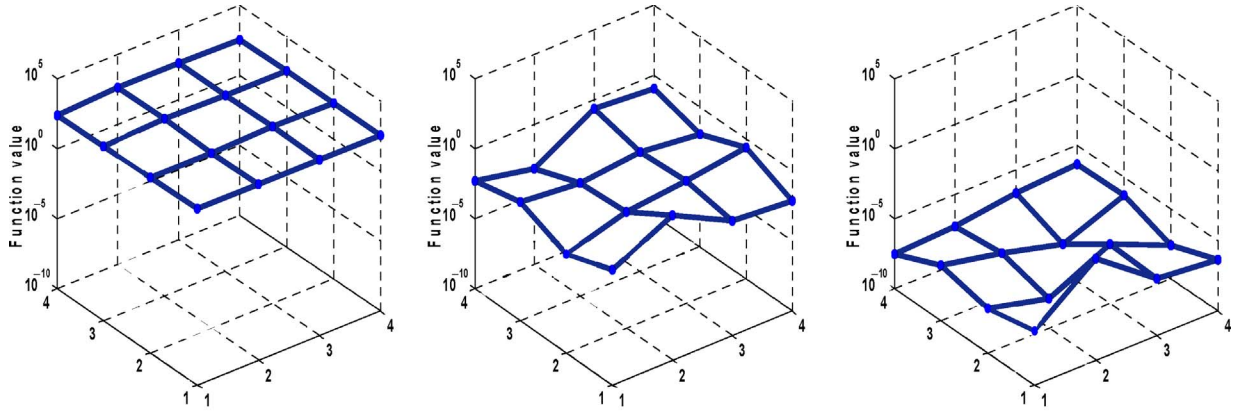

Fig. 3. 10-D function optimization example by SOPFN with a 4 × 4 network. (a) Initialization state. (b) 50th generation state. (c) 100th generation state.

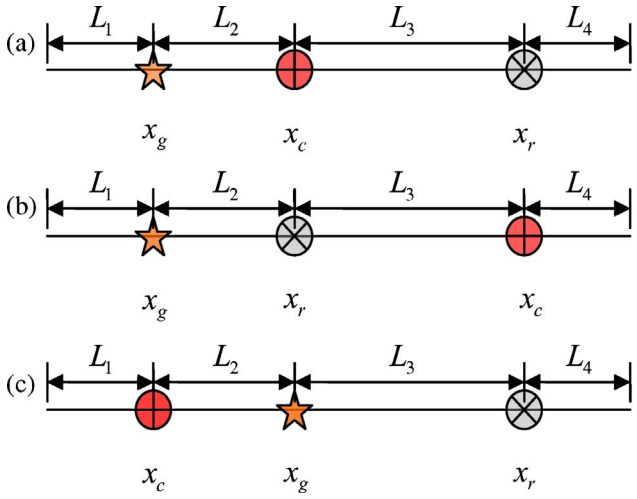

Fig. 4. 1-D search space with target and obstacle neurons. (a) *Case I*: $x_g < x_c < x_r$. (b) *Case II*: $x_g < x_r < x_c$. (c) *Case III*: $x_c < x_g < x_r$. x_c denotes best candidate value, x_r denotes worst candidate value, and x_g denotes global optimum.

B. Self-Organizing Potential Field Network Algorithm

An optimization problem can be expressed in a form of finding the vector x, where $x \in \Re^D$, through minimizing the cost function $f(x)$ with constraints of $g(x) \leq 0$ and $x_l \leq x \leq x_u$. In this paper, we assume that function variables are continuous and are floating points. The detailed optimization procedures of the SOPFN algorithm are as follows.

1) *Initialization:* Randomize the initial weights of $M \times N$ neurons. The weights satisfy the uniform distribution and optimization constraints.
2) *Construction of the Potential Field:* Find the target neuron c with the best solution and the obstacle neuron r with the worst solution according to

$$c = \arg\min_i (f(w_i)) \quad i \in \{1, 2, \ldots, M \times N\} \tag{13}$$

$$r = \arg\max_i (f(w_i)) \quad i \in \{1, 2, \ldots, M \times N\}. \tag{14}$$

3) *1-D Weight Updating:* For every neuron i, randomly choose an integer $k \in [1, D]$.
   a) Update the kth component of the weight vector of neuron i toward the target neuron c. Generate a set of weights w_i' according to the following rule:

$$w_{ij}' = \begin{cases} w_{ij} + \zeta \cdot F_{\text{att}}^{ij}, & \text{if } j = k \text{ and } i \in N_{\text{set}} \\ w_{ij}, & \text{otherwise} \end{cases} \quad j=1, 2, \ldots, D \tag{15}$$

   where $\zeta \in \{\Delta, 2\Delta, \ldots, m\Delta\}$, Δ is the step size, m is the step number, and N_{set} is the neighborhood set of neuron c.
   b) Update the kth component of each weight vector of w_i' away from the obstacle neuron r. Generate another set of weights w_i'' according to the following rule:

$$w_{ij}'' = \begin{cases} w_{ij}' - F_{\text{rep}}^{ij}, & \text{if j} = k \& i \in N_{\text{set}} \\ w_{ij}', & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, D. \tag{16}$$

   c) Find the best solution between w_i' and w_i'', and denote it as the new weight of neuron i as following:

$$w_i = \arg\min_{[w_i', w_i'']} \{f(w_i'), f(w_i'')\}. \tag{17}$$

4) *Self-adaptation:* Compare function values to reassign the target neuron c and obstacle neuron r among all neurons.
5) If the specified maximum number of iteration is reached or the stopping criteria are satisfied, stop running. Otherwise, go to step 3.

Step 2, which finds the target neuron and obstacle neuron, represents the competitive behavior of SOPFN. Step 3, which generates new weights of neurons, represents the cooperative behavior of SOPFN. The step size Δ defines the sample granularity of the attractive force in the search space. A large value of Δ can speed up the search, while a small value of Δ can increase the diversity. The step number m, which is larger than 1, defines the maximum updating boundary. When m increases, the diversity is increased as well. But m cannot be

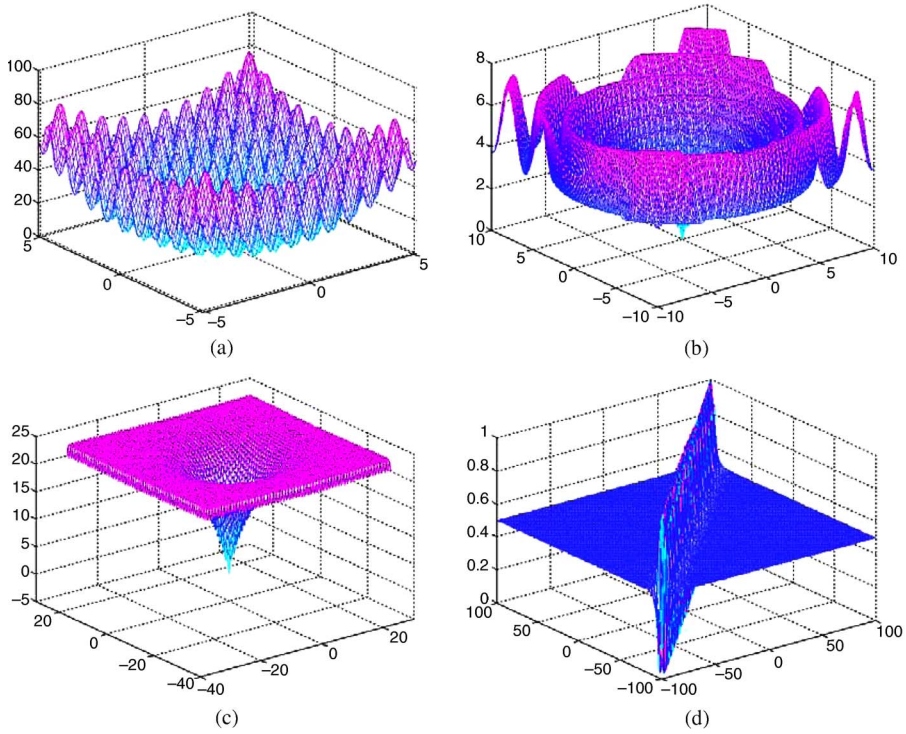

Fig. 5. Landscape maps of four 2-D functions. (a) Rastrigin's function. (b) Stretched V sine wave function. (c) Ackley's function. (d) Pathological function.

too large because of the subsequent increase of computational complexity. The learning rates α_1 and α_2 range from 0 to 1. For the attractive learning rate α_1, it has an effect on the convergence rate. The neighborhood radius σ is the integer ranged from one to the map size. The number of neurons $M \times N$ is determined by the nature of the problem. Usually a small value is used for a low-dimensional or unimodal problem, and a large value is used for a high-dimensional or multimodal problem.

Fig. 3 illustrates a 4×4 SOPFN network working on optimizing the function $f(x) = \sum_{i=1}^{D} |x_i|$ $(d = 10)$ whose optimum value is 0. The initial 16 neuron weights are randomized in the range of $[-100\ 100]$, and the corresponding function values are shown in Fig. 3(a). The average function value is 355.39. After 50 iterations, all the neuron weights are optimized by SOPFN, and the average function value decreases to 0.18 as shown in Fig. 3(b). After 100 iterations, the function values of all neurons are shown in Fig. 3(c) with the average value 1.17×10^{-4}.

C. Cooperative and Competitive Behaviors of SOPFN

This section studies the cooperative and competitive behaviors of the SOPFN algorithm. In the cooperative behavior, since the parameter ζ spreads over a range of values, SOPFN generates a series of candidate solutions for a neuron. As a result, the diversity is increased. The competitive behavior of SOPFN enables neurons not only to move toward the best solution, but also away from the worst solution. This mechanism is able to prevent SOPFN from premature convergence to a great extent and make neurons escape from the local optimum with a larger probability. The results presented in Section IV corroborate the contribution of SOPFN.

To compare the traditional evolutionary algorithm with SOPFN, we use an example of 1-D search space related to one component of a neuron weight vector. Three cases are illustrated in Fig. 4, where x_c, x_r, and x_g are the component values of the best candidate solution, the worst candidate solution, and the global optimum solution, respectively. In each case, there are four possible regions labeled as L_1, L_2, L_3, and L_4 that the corresponding component value of a neuron x_p can be subjected to.

Table I demonstrates the effects of the potential forces. In the traditional EA algorithm, only one force, the attractive force F_{att}, makes x_p move toward x_c. In the SOPFN algorithm, in addition to the attractive force F_{att}, the repulsive force F_{rep} pulls x_p away from x_r. For example, in case I for EA, F_{att} pushes x_p away from the optimum x_g, when x_p is subjected to L_2. But for SOPFN, F_{rep} pushes x_p toward x_g, making the distance between x_g and x_p smaller than that of EA. When x_p is subjected to L_3, F_{att}, and F_{rep} of SOPFN both push x_p toward x_g. So x_p is closer to x_g in SOPFN. When x_p is subjected to L_1 or L_4, F_{rep} pulls x_p away from x_g. According to step 3(c) of the SOPFN algorithm, only the attractive force is considered, which means $F_{\text{rep}} = 0$. Overall, Table I shows that the repulsive force pushes the weight of the neuron closer to the global optimum value in the four shaded regions out of the total 12 regions. This operation can reduce the searching time required by neurons to reach the optimum value.

IV. Simulations and Results

The performance of SOPFN is compared with the SA, SOSENs, PSO, CLPSO, CPSO-S_K, and SOMA algorithms on eight benchmark functions. These studied functions include

TABLE I
COMPARISONS BETWEEN TRADITIONAL EA AND SOPFN IN 1-D SEARCH SPACE

	Case I		Case II		Case III	
	Traditional EA	SOPFN	Traditional EA	SOPFN	Traditional EA	SOPFN
L_1	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$
L_2	$F_{att}(-)$	$F_{att}(-)+F_{rep}(+)$	$F_{att}(-)$	$F_{att}(-)+F_{rep}(+)$	$F_{att}(-)$	$F_{att}(-)$
L_3	$F_{att}(+)$	$F_{att}(+)+F_{rep}(+)$	$F_{att}(-)$	$F_{att}(-)$	$F_{att}(+)$	$F_{att}(+)+F_{rep}(+)$
L_4	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$	$F_{att}(+)$

"(+)" means that the neuron moves toward the optimum value by the force.
"(−)" means that the neuron moves away from the optimum value by the force.

unimodal (the first four functions) and multimodal (the last four functions) problems that are widely used for measuring the optimization performance. The formulas of functions are shown as follows.

1) Sphere function

$$f_1(x) = \sum_{i=1}^{D} x_i^2. \tag{18}$$

2) Rosenbrock's function

$$f_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2). \tag{19}$$

3) Third De Jong function

$$f_3(x) = \sum_{i=1}^{D} |x_i| . \tag{20}$$

4) Fourth De Jong function

$$f_4(x) = \sum_{i=1}^{D} i x_i^4. \tag{21}$$

5) Rastrigin's function

$$f_5(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10). \tag{22}$$

6) Stretched V sine wave function

$$f_6(x) = \sum_{i=1}^{D-1} (x_i^2 + x_{i+1}^2)^{0.25} (1 + \sin(50(x_i^2 + x_{i+1}^2)^{0.1})^2). \tag{23}$$

7) Ackley's function

$$f_7(x) = \sum_{i=1}^{D-1} (20 + e - 20e^{-0.2\sqrt{0.5(x_i^2 + x_{i+1}^2)}} - e^{0.5(\cos(2\pi x_i) + \cos(2\pi x_{i+1}))}). \tag{24}$$

8) Pathological function

$$f_8(x) = \sum_{i=1}^{D-1} \left(0.5 + \frac{\sin\left(\sqrt{100x_i^2 + x_{i+1}^2}\right)^2 - 0.5}{1 + 0.001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2} \right). \tag{25}$$

The landscape maps of Rastrigin, Stretched V sine wave, Ackley and Pathological functions with two variables are shown in Fig. 5. All eight functions are studied on 30 and 100 dimensions respectively. In this paper, we use MATLAB to conduct all simulations, and each of them is conducted 20 times. Table II lists the optimum solution x^*, the optimum function value $f(x^*)$, the initialization range, the threshold (experimental optimum function value) and the maximum number of iteration for each function.

The population size is set to 25 in PSO, CLPSO, CPSO-S_K, and SOMA. The network size is set to 5×5 in SOSENs and SOPFN. In SOMA, *PRT* is set to 0.3, Δ and k are set to 0.11 and 27, respectively. In PSO, ω is set to 0.72, c_1 and c_2 are set to 1.49 [6]. In CLPSO, the refreshing gap is set to 7. In CPSO-S_K, the split factor K is set to 5. In SA, the temperature decreases by $T := \varepsilon \cdot T$, where ε is set to 0.99. In SOSENs, α is set to 1. In SOPFN, Δ and m are set to 1 and 3, respectively, α_1 and α_2 are set to 0.3, and σ is set to 3.

To illustrate the effects of these parameters in SOPFN, we examine the comparative performance on the 30-D 4th De Jong function f_4. Fig. 6 shows the optimization results with different values of α_1, α_2, σ, Δ, and m. Fig. 6(b) indicates that α_2 is the less sensitive parameter due to the characteristic that there is no significant difference of the function value in different values of α_2. Also, Fig. 6(c) and (e) demonstrates that σ and m are less sensitive when they are large.

A. Simulation Results of 30-D Functions

The performances of the seven algorithms with minimum function values, mean function values, and the numbers of successful runs out of 20 runs on eight test functions are presented in Table III. The best results among the seven algorithms are shown in bold. Fig. 7 illustrates the mean function value at every generation for each algorithm. The results indicate that the numbers of the aforementioned functions that SA, SOSENs, PSO, CLPSO, CPSO-S_5, SOMA, and SOPFN can reach the thresholds are 0, 3, 1, 1, 4, 2, and 6, respectively.

Among these seven algorithms, SOPFN evidently surpasses the other six algorithms on f_5, f_6, and f_7. These functions are multimodal with numerous local minima as shown in Fig. 5. The results indicate that SOPFN is able to deliver better performance by avoiding being trapped in local minima. For the unimodal functions f_1, f_3, and f_4, SOPFN obtains the experimental optimum values, but it does not have the fastest convergence rate as shown in Fig. 7. For the Rosenbrock's function f_2 whose global minimum is inside a long, narrow, parabolic shaped flat valley, SOPFN does not attain the optimum value. And all the other algorithms do not perform well except CPSO-S_5. In [30], it states that an algorithm may not produce the same high level of result on all different classes

TABLE II
EXPERIMENTAL PARAMETERS OF EIGHT BENCHMARK FUNCTIONS

f	x^*	$f(x^*)$	Initialization Space	Threshold		Maximum Number of Iteration
				30-D	100-D	
f_1	$[0, 0, \ldots, 0]$	0	$[-5.12, 5.11]^D$	10^{-50}	10^{-20}	3000
f_2	$[1, 1, \ldots, 1]$	0	$[-2.048, 2.047]^D$	10^{-2}	10^{0}	3000
f_3	$[0, 0, \ldots, 0]$	0	$[-2.048, 2.047]^D$	10^{-20}	10^{-10}	3000
f_4	$[0, 0, \ldots, 0]$	0	$[-1.28, 1.27]^D$	10^{-20}	10^{-10}	3000
f_5	$[0, 0, \ldots, 0]$	0	$[-5.12, 5.11]^D$	10^{+1}	10^{+2}	3000
f_6	$[0, 0, \ldots, 0]$	0	$[-10, 10]^D$	10^{-2}	10^{0}	3000
f_7	$[0, 0, \ldots, 0]$	-8.88×10^{-16}	$[-30, 30]^D$	10^{-2}	10^{0}	3000
f_8	$\left[\frac{k\pi}{\sqrt{101}}, \frac{k\pi}{\sqrt{101}}, \ldots, \frac{k\pi}{\sqrt{101}}\right]$ $-320 < k < 320$, k *is an integer*	0	$[-100, 100]^D$	10^{0}	10^{+1}	3000

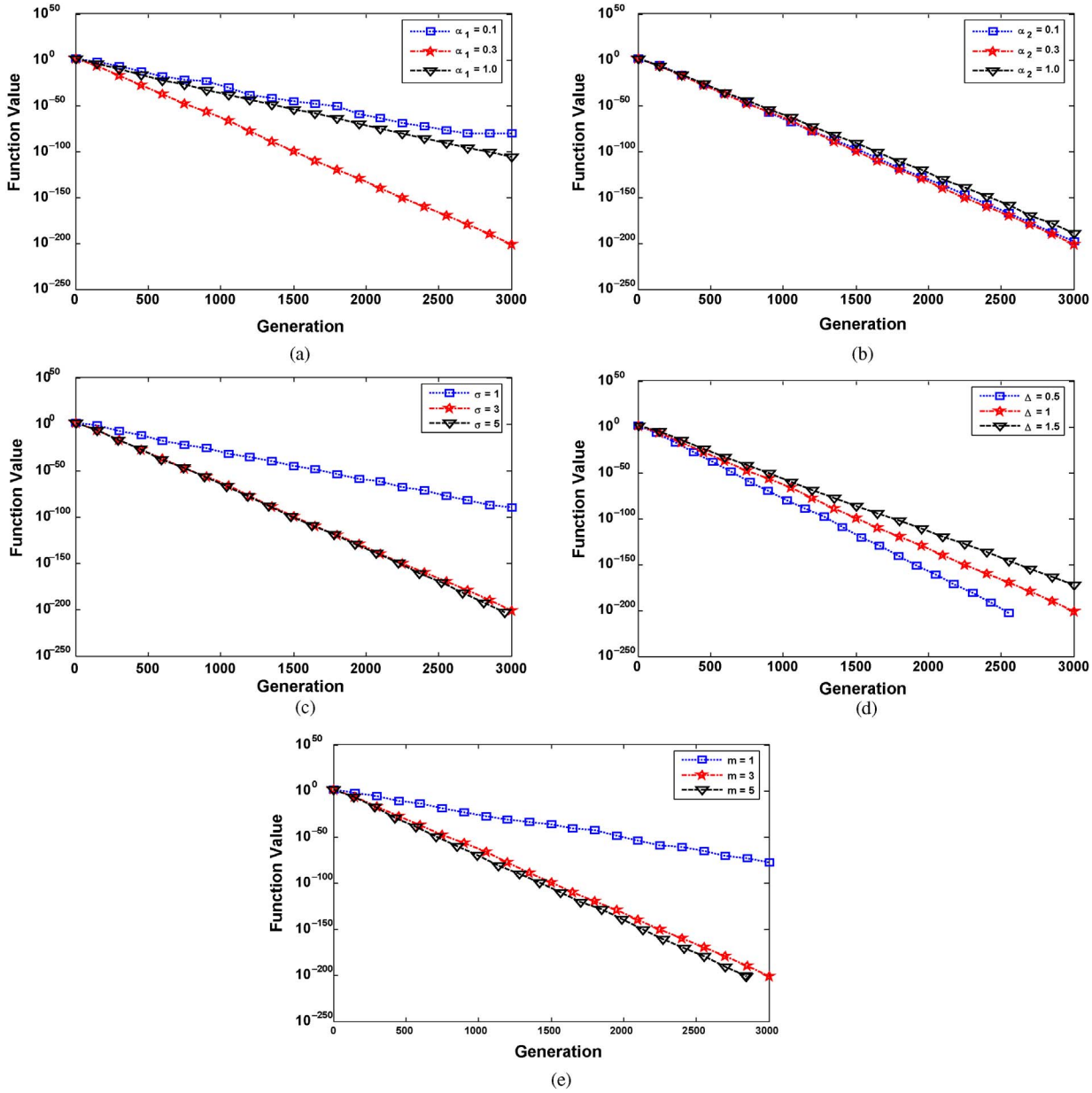

Fig. 6. Parameter comparisons of SOPFN on f_4. (a) α_1. (b) α_2. (c) σ. (d) Δ. (e) m.

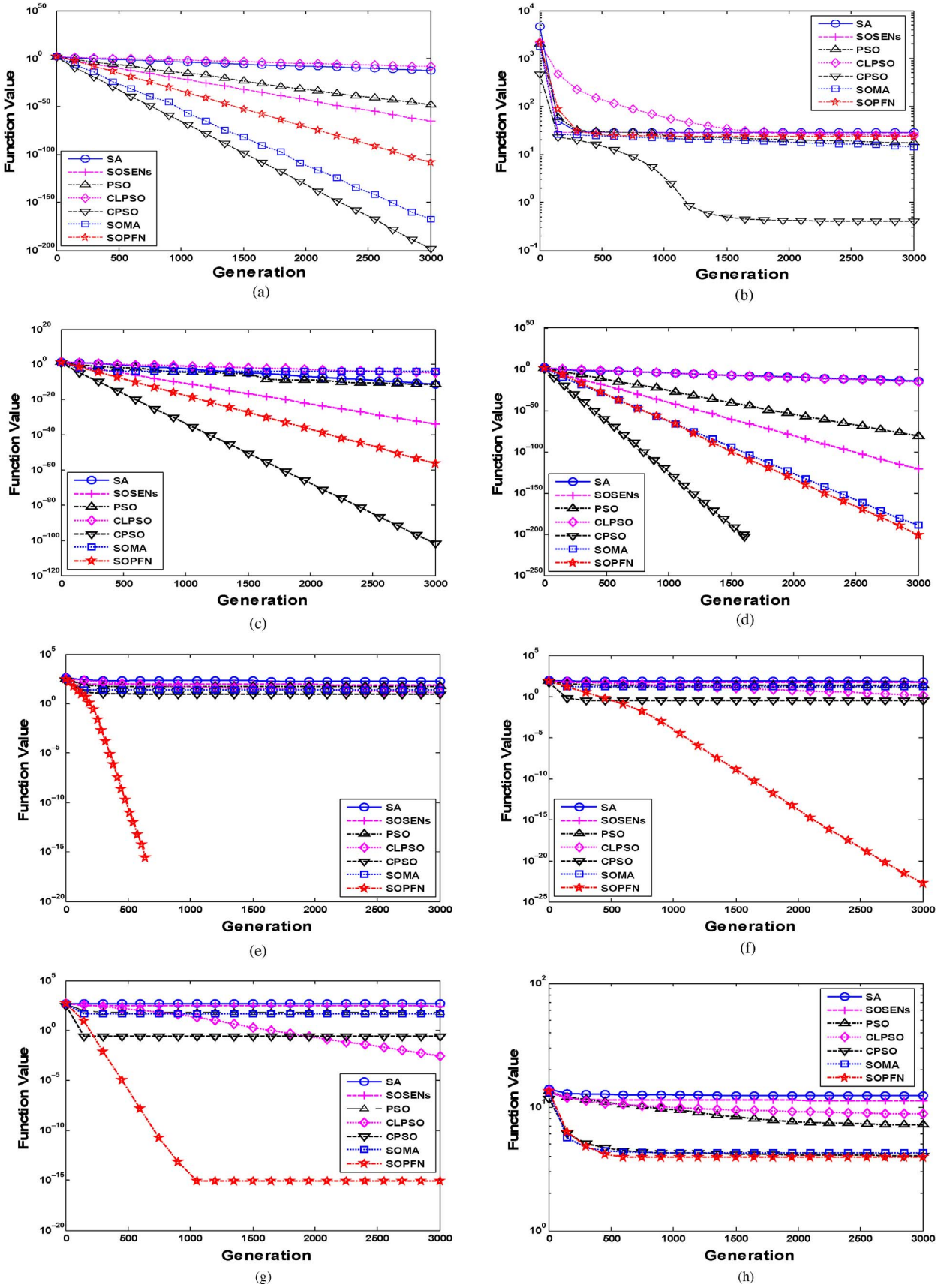

Fig. 7. Mean function value profiles on 30-D functions. (a) Sphere function. (b) Rosenbrock's function. (c) Third De Jong function. (d) Fourth De Jong function. (e) Rastrigin's function. (f) Stretched V sine wave function. (g) Ackley's function. (h) Pathological function.

TABLE III
COMPARATIVE PERFORMANCES OF SEVEN ALGORITHMS ON 30-D FUNCTIONS

Function 1	SA	SOSENs	PSO	CLPSO	CPSO-S_5	SOMA	SOPFN
Mean	4.01e−013	6.42e−066	5.23e−049	3.65e−009	**6.36e−199**	5.83e−168	4.65e−109
Minimum	1.60e−013	2.62e−067	1.71e−058	1.32e−009	7.07e−201	1.54e−174	2.02e−113
Number of successful runs	0	20	18	0	20	20	20
Function 2							
Mean	28.75	28.57	17.38	24.27	**0.40**	14.55	23.91
Minimum	27.22	26.12	1.85	21.14	4.57e−004	12.66	18.57
Number of successful runs	0	0	0	0	17	0	0
Function 3							
Mean	6.48e−012	9.85e−035	7.02e−012	1.45e−005	**2.04e−102**	1.63e−004	5.26e−057
Minimum	4.82e−012	4.13e−035	5.74e−024	8.24e−006	2.77e−104	1.41e−057	4.03e−058
Number of successful runs	0	20	1	0	20	8	20
Function 4							
Mean	3.92e−015	1.20e−121	6.46e−082	3.01e−016	6.68e−201	4.42e−190	**4.13e−201**
Minimum	7.97e−018	4.21e−124	1.04e−090	6.26e−017	2.45e−202	1.17e−198	6.70e−203
Number of successful runs	0	20	20	0	20	20	20
Function 5							
Mean	189.62	83.32	56.86	14.57	9.20	22.44	**0**
Minimum	119.27	50.44	35.82	9.08	3.98	8.95	0
Number of successful runs	0	0	0	1	13	1	20
Function 6							
Mean	69.46	46.43	24.10	1.43	0.33	14.54	**1.90e−023**
Minimum	55.05	40.11	13.27	1.03	4.27e−050	5.92	1.12e−025
Number of successful runs	0	0	0	0	8	0	20
Function 7							
Mean	469.47	276.26	64.72	0.0027	0.26	43.86	**−8.88e−016**
Minimum	407.63	246.28	30.13	0.0019	−8.88e−016	5.16	−8.88e−016
Number of successful runs	0	0	0	20	19	0	20
Function 8							
Mean	12.32	11.23	7.17	8.76	4.05	4.24	**3.91**
Minimum	11.71	10.56	5.02	8.14	2.52	3.11	2.79
Number of successful runs	0	0	0	0	0	0	0

of problems. In this paper, we also find the SOPFN algorithm more suitable to the complex multimodal problems, especially with numerous local minima.

B. Simulation Results of 100-D Functions

The performances of the seven algorithms on 100-D test functions are presented in Table IV and Fig. 8. The results show that SA, SOSENs, PSO, CLPSO, CPSO-S_5, SOMA, and SOPFN can reach the thresholds in 1, 3, 0, 0, 3, 0, and 6 function cases, respectively. The results demonstrate that SOPFN achieves the best results on all the multimodal functions f_5, f_6, f_7, and f_8 among the seven algorithms. The conclusion, which is identical with that of 30-D function, indicates that the SOPFN algorithm is more effective on the multimodal problems.

V. ANALYSIS OF SOPFN ALGORITHM

The performance of an algorithm is evaluated in terms of three criteria—accuracy, convergence rate, and robustness [34]. The accuracy is measured by how close it is between the algorithm's mean function value and the global optimum. According to the results listed in Table III, SOPFN delivers the closest values to the global optima in five functions. Among these five solutions, three of them are found to be superior to those by the other algorithms. The convergence rate of an algorithm is evaluated by the number of generations required to reach the global optimum. Fig. 7(e)–(g) shows that SOPFN has the distinctly fastest convergence rate compared with other algorithms. The results also indicate that SOPFN exhibits the best convergence characteristic on multimodal functions. The robustness is evaluated by the number of successful runs out of the total runs. A successful run means that its optimization result is less than the given threshold. The larger the number of successful runs is, the more robust the algorithm is. The results summarized in Table III indicate that SOPFN can always reach the thresholds in the total 20 runs on f_1, f_3, f_4, f_5, f_6, and f_7. Among all the studied algorithms, we can conclude that SOPFN is the most robust algorithm.

It is interesting to study the effect of omitting step 3(b), i.e., there is no repulsive force. Fig. 9 and Table V show the comparative results between SOPFN and SOPFN-R (without the repulsive force) on eight 30-D functions. The results show that SOPFN outperforms SOPFN-R on all functions except the Ackley's function f_7, in which SOPFN and SOPFN-R deliver the same result.

Another interesting study is to investigate the effect of the $1 - d$ weight updating strategy of the SOPFN algorithm which is used to solve the problem of "two steps forward, one step back." Fig. 10 and Table VI show the results of

TABLE IV
COMPARATIVE PERFORMANCES OF SEVEN ALGORITHMS ON 100-D FUNCTIONS

Function 1	SA	SOSENs	PSO	CLPSO	CPSO-S_5	SOMA	SOPFN
Mean	2.28e−012	9.45e−036	0.17	0.02	**4.36e−086**	0.05	1.13e−027
Minimum	1.49e−012	3.10e−036	1.95e−005	0.01	7.39e−089	1.31e−012	6.84e−029
Number of successful runs	0	20	0	0	20	0	20
Function 2							
Mean	98.18	97.92	148.38	111.53	**85.90**	97.28	95.23
Minimum	95.74	95.34	93.37	99.50	44.85	88.09	92.87
Number of successful runs	0	0	0	0	0	0	0
Function 3							
Mean	2.48e−010	2.46e−019	0.97	0.15	**4.47e−025**	1.31	1.35e−014
Minimum	1.54e−010	1.69e−019	0.08	0.12	6.24e−035	0.38	5.22e−015
Number of successful runs	0	20	0	0	20	0	20
Function 4							
Mean	5.63e−014	1.49e−062	3.10e−008	1.27e−004	**3.81e−139**	3.18e−005	1.24e−050
Minimum	1.20e−015	1.53e−063	4.11e−011	5.30e−005	1.84e−144	6.48e−043	6.43e−054
Number of successful runs	20	20	1	0	20	16	20
Function 5							
Mean	823.70	388.69	280.20	405.49	137.95	130.32	**0**
Minimum	683.74	271.62	213.92	352.89	111.44	97.51	0
Number of successful runs	0	0	0	0	0	2	20
Function 6							
Mean	269.87	180.74	161.30	53.98	45.64	106.92	**0.015**
Minimum	245.32	168.84	133.87	48.62	32.16	95.22	0.0057
Number of successful runs	0	0	0	0	0	0	20
Function 7							
Mean	1683.90	783.30	726.53	23.41	99.70	537.71	**1.94e−012**
Minimum	1510.90	672.00	542.54	15.59	25.80	402.07	8.77e−013
Number of successful runs	0	0	0	0	0	0	20
Function 8							
Mean	46.01	43.99	34.49	41.83	22.12	26.81	**15.97**
Minimum	44.83	42.73	30.56	41.32	19.76	23.99	13.69
Number of successful runs	0	0	0	0	0	0	0

TABLE V
COMPARATIVE PERFORMANCES BETWEEN SOPFN AND SOPFN-R ON 30-D FUNCTIONS

Algorithm / Function	SOPFN v_1	SOPFN-R v_2	Ratio v_2/v_1
Function 1	4.65e−109	1.42e−086	3.05×10^{22}
Function 2	23.91	27.13	1.13
Function 3	5.26e−057	1.34e−045	2.54×10^{11}
Function 4	4.13e−201	2.48e−167	6.00×10^{33}
Function 5	0	0.10	∞
Function 6	1.90e−023	1.08e−019	5.69×10^{3}
Function 7	−8.88e−016	−8.88e−016	1
Function 8	3.91	5.04	1.29

SOPFN with different numbers of updated components d on four 30-D functions. On Rosenbrock's function f_2 and 4th De Jong function f_4, SOPFN achieves the best values when $d = 1$. On Sphere function f_1 and Pathological function f_8, SOPFN obtains the best values when $d = 5$ and $d = 2$, respectively. It is also noticed that using a larger number of updated components has no significant effect on improving the accuracy, but increases the computational cost. Thus, the $1 - d$ updating strategy appears to be the optimal choice for SOPFN.

For comparing the convergence performance among population-based algorithms, Table VII demonstrates the numbers of generations required to reach the specified stopping criteria of eight 100-D functions for the SOSENs, PSO, CLPSO, CPSO-S_5, SOMA, and SOPFN algorithms. The results show that SOPFN can reach all eight stopping criteria within 100 000 generations, while other algorithms fail to converge on some functions. The average computational time per 100 generations for each algorithm is also detailed in Table VII. The results show that the first three algorithms with the fastest computational time per generation are CLPSO, PSO, and SOPFN with 0.16 s, 0.17 s, and 0.96 s, respectively. The SOMA algorithm, which requires 4.38 s, is found to be the most computationally demanding algorithm. Although SOPFN is relatively more computationally complex than CLPSO and PSO in each generation, SOPFN exhibits the best convergence performance in terms of reaching the stopping criterion. In the case of f_5, the converged values of CLPSO and PSO are 97.41 and 280.20, respectively, which are much larger than the converged value 0.001 obtained by SOPFN. Table VII illustrates that SOPFN delivers superior optimization values on the eight functions.

The computational complexity of a population-based algorithm can be defined as $O(P \cdot D \cdot M)$, where P is the number of neurons, M is the number of new generated weights for

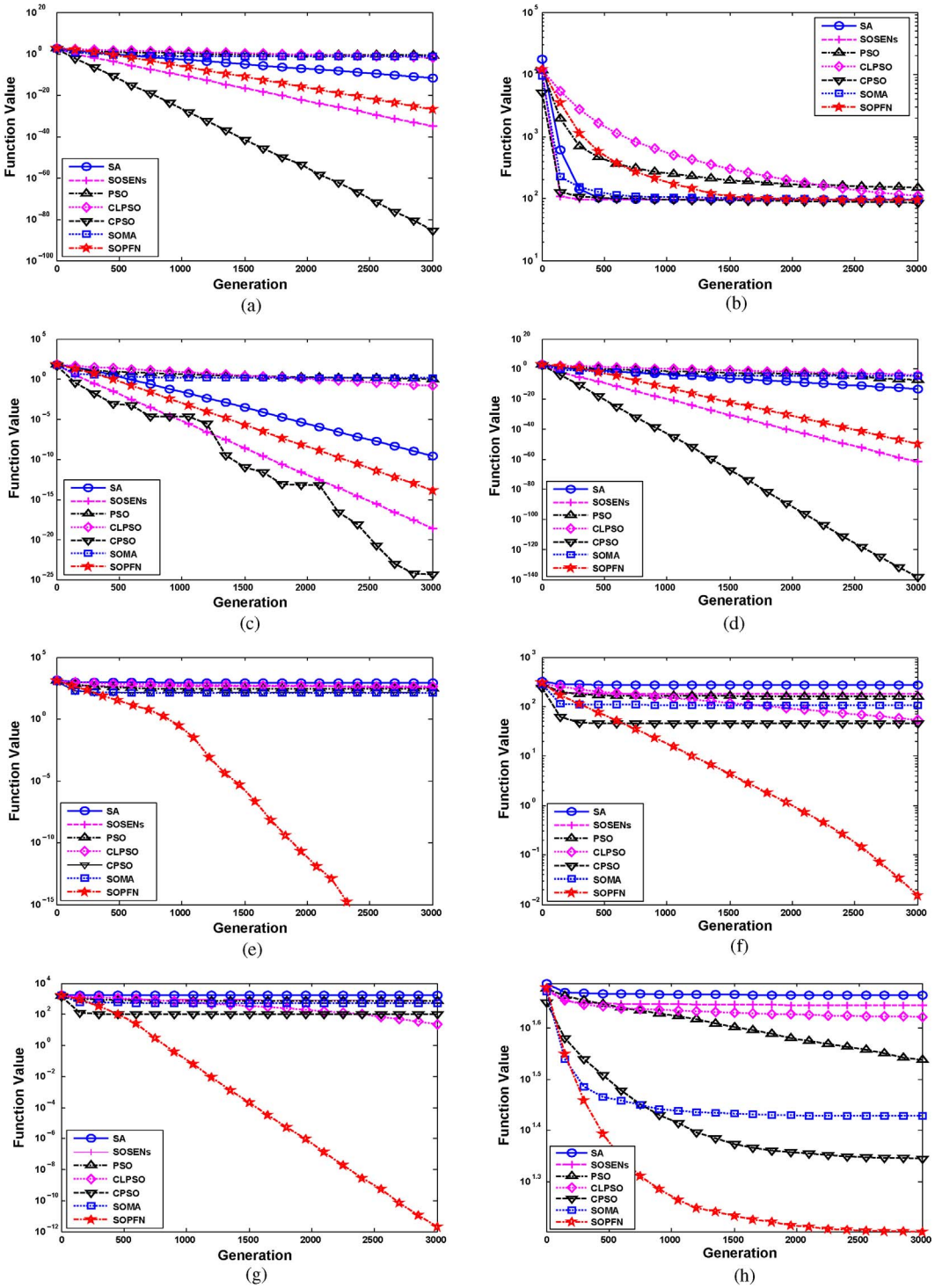

Fig. 8. Mean function value profiles on 100-D functions. (a) Sphere function. (b) Rosenbrock's function. (c) Third De Jong function. (d) Fourth De Jong function. (e) Rastrigin's function. (f) Stretched V sine wave function. (g) Ackley's function. (h) Pathological function.

TABLE VI

COMPARATIVE PERFORMANCES OF SOPFN WITH DIFFERENT NUMBERS OF UPDATED COMPONENTS ON 30-D FUNCTIONS

Algorithm / Function	SOPFN $d = 1$	SOPFN $d = 2$	SOPFN $d = 5$	SOPFN $d = 10$	SOPFN $d = 30$
Function 1	4.65e−109	8.57e−111	4.89e−122	1.68e−115	5.39
Function 2	23.91	25.49	26.72	35.33	105.07
Function 4	4.13e−201	1.32e−183	7.29e−142	4.01e−154	0.12
Function 8	3.91	2.83	3.65	5.62	12.25

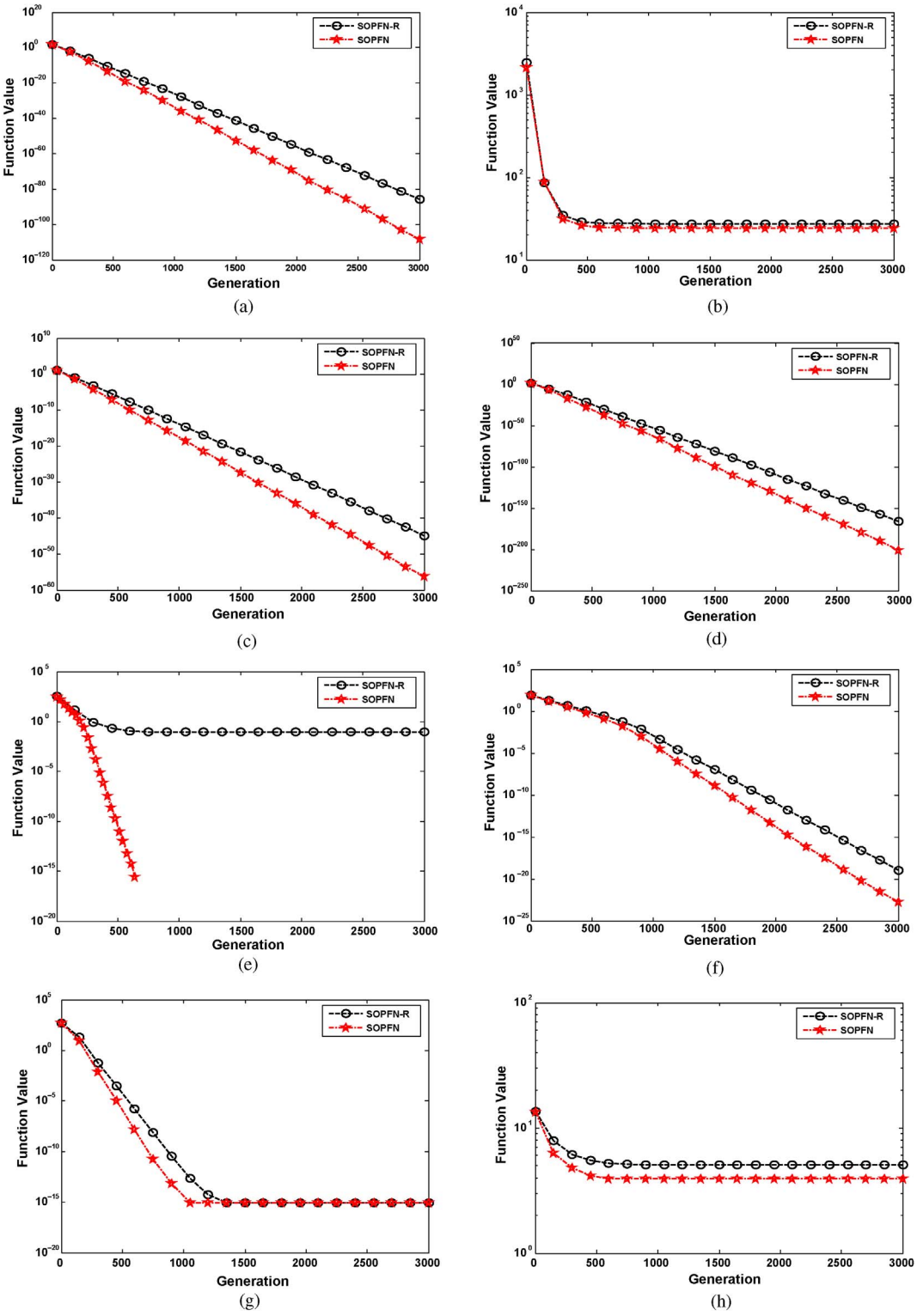

Fig. 9. Mean 30-D function value profiles of SOPFN and SOPFN-R. (a) Sphere function. (b) Rosenbrock's function. (c) Third De Jong function. (d) Fourth De Jong function. (e) Rastrigin's function. (f) Stretched V sine wave function. (g) Ackley's function. (h) Pathological function.

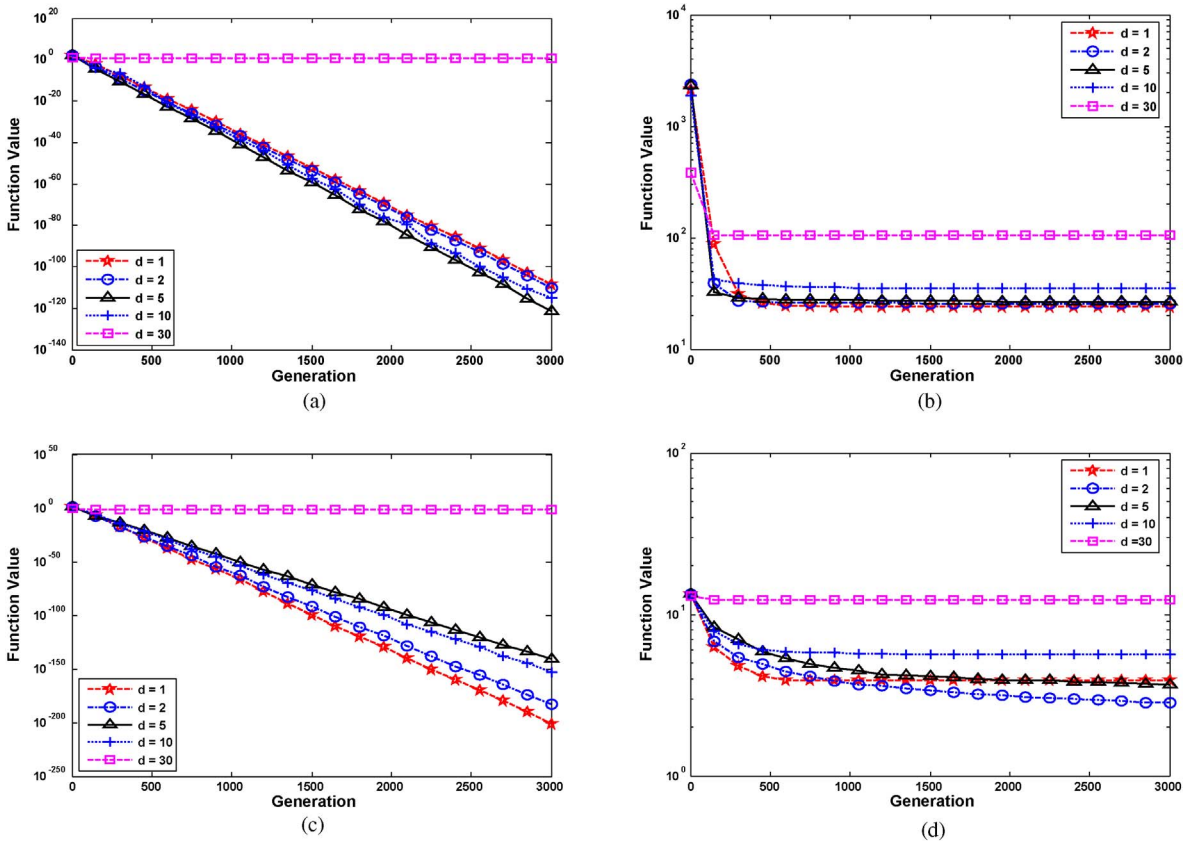

Fig. 10. Mean 30-D function profiles of SOPFN with different numbers of updated components. (a) Sphere function. (b) Rosenbrock's function. (c) Fourth De Jong function. (d) Pathological function.

TABLE VII
COMPARATIVE PERFORMANCES OF CONVERGENCE FOR SIX POPULATION-BASED ALGORITHMS ON 100-D FUNCTIONS

Algorithm / Function	Stopping Criterion	Number of Generations Required to Reach the Stopping Criterion					
		SOSENs	PSO	CLPSO	CPSO-S_5	SOMA	SOPFN
f_1	10^{-20}	1778	14 019	16 507	799	×	2364
f_2	10^{2}	219	19 475	4592	1253	819	1440
f_3	10^{-20}	3223	39 200	25 779	1928	×	4051
f_4	10^{-20}	1004	8205	11 641	499	4587	1310
f_5	10^{-3}	×	×	×	×	×	1157
f_6	10^{0}	×	×	9530	×	×	1916
f_7	10^{-10}	×	×	17 288	×	×	2608
f_8	20	×	×	×	×	×	696
Average computational time per 100 generations (s)		1.15	0.17	0.16	1.98	4.38	0.96

"×" means the optimization result cannot reach the stopping criterion at 100 000 generations.

a neuron at each generation, and D is the number of updated components in the weight vector. Because of the $1-d$ updating strategy, SOPFN has the less computational complexity, when the function dimensionality is large.

VI. CONCLUSION

SOPFN is a new evolutionary algorithm that models the search space as a self-organizing potential field. In SOPFN, the neuron with the best solution is considered as the target with the attractive force, while the neuron with the worst solution is considered as the obstacle with the repulsive force. SOPFN can be considered as a combination of two behaviors, competitive and cooperative. In the competitive behavior, the target and obstacle neurons are found for neuron training. This mechanism speeds up the convergence rate and increases the probability of escaping from the local optimum. In the cooperative behavior, the winner's neighboring neurons are updated to generate new weights at each generation. This mechanism increases the diversity.

The presented results and analysis demonstrate that SOPFN has remarkable performance on multimodal problems, especially for those with numerous local optima. For the case of unimodal problems, SOPFN is not superior in the convergence

rate. Lastly, based on the algorithm evaluation, SOPFN is an effective and robust optimization algorithm.

ACKNOWLEDGMENT

The authors would like to thank the editors and anonymous reviewers for providing valuable comments and suggestions.

REFERENCES

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

[2] E. Berglund and J. Sitte, "The parameterless self-organizing map algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 305–316, Mar. 2006.

[3] H. G. Beyer, *The Theory of Evolution Strategies*. Berlin, Germany: Springer-Verlag, 2001.

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

[5] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, Oct. 1995, pp. 39–43.

[6] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. Congr. Evol. Comput.*, San Diego, CA, Jul. 2000, pp. 84–89.

[7] F. G. Guimaraes, F. Campelo, H. Igarashi, D. A. Lowther, and J. A. Ramirez, "Optimization of cost functions using evolutionary algorithms with local learning and local search," *IEEE Trans. Magn.*, vol. 43, no. 4, pp. 1641–1644, Apr. 2007.

[8] Z. Hao, G. Guo, and H. Huang, "A particle swarm optimization algorithm with differential evolution," in *Proc. 6th Int. Conf. Mach. Learn. Cybern.*, vol. 2. Hong Kong, Aug. 2007, pp. 1031–1035.

[9] J. Huhse, T. Villmann, P. Merz, and A. Zell, "Evolution strategy with neighborhood attraction using a neural gas approach," in *Proc. 7th Int. Conf. Parallel Problem Solving Nature*, LNCS 2439. Jan. 2002, pp. 391–400.

[10] T. Kaji, "Approach by ant tabu agents for traveling salesman problem," in *Proc. IEEE Int. Conf. Systems, Man, Cybern.*, Tucson, AZ, Oct. 2001, pp. 3429–3434.

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

[12] H. K. Kim, J. K. Chong, K. Y. Park, and D. A. Lowther, "Differential evolution strategy for constrained global optimization and application to practical engineering problems," *IEEE Trans. Magn.*, vol. 43, no. 4, pp. 1565–1568, Apr. 2007.

[13] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[14] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer-Verlag, 1997.

[15] R. Kumar and P. Rockett, "Multiobjective genetic algorithm partitioning for hierarchical learning of high-dimensional pattern spaces: A learning-follows-decomposition strategy," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 822–830, Sep. 1998.

[16] J. Lampinen and R. Storn, "Differential evolution," in *New Optimization Techniques in Engineering*, G. C. Onwubolu and B. V. Babu, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 123–166.

[17] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[18] J. J. Liang and P. N. Suganthan, "Adaptive comprehensive learning particle swarm optimizer with history learning," in *Proc. 6th Int. Conf. Simul. Evol. Learn.*, LNCS 4247. Oct. 2006, pp. 213–220.

[19] T. Maruyama and H. Igarashi, "An effective robust optimization based on genetic algorithm," *IEEE Trans. Magn.*, vol. 44, no. 6, pp. 990–993, Jun. 2008.

[20] A. A. Masoud, "Decentralized self-organizing, potential field-based control for individually motivated mobile agents in a cluttered environment: A vector-harmonic potential field approach," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 3, pp. 372–390, May 2007.

[21] A. A. Masoud and M. M. Bayoumi, "Robot navigation using the vector potential approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1. Atlanta, GA, May 1993, pp. 805–811.

[22] S. A. Masoud and A. A. Masoud, "Constrained motion control using vector potential fields," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 30, no. 3, pp. 251–272, May 2000.

[23] M. Milano, P. Koumoutsakos, and J. Schmidhuber, "Self-organizing nets for optimization," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 758–765, May 2004.

[24] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. 3rd Conf. Parallel Problem Solving Nature*, LNCS 866. 1994, pp. 249–257.

[25] R. Senkerik, I. Zelinka, and D. Davendra, "Comparison of evolutionary algorithms in the task of chaos control optimization," in *Proc. IEEE Congr. Evolut. Comput.*, Singapore, Sep. 2007, pp. 3952–3958.

[26] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evolut. Comput.*, Anchorage, AK, May 1998, pp. 69–73.

[27] N. C. Tsourveloudis, K. P. Valavanis, and T. Hebert, "Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 490–497, Aug. 2001.

[28] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 586–600, May 2000.

[29] M. D. Vose, *Simple Genetic Algorithm: Foundation and Theory*. Cambridge, MA: MIT Press, 1999.

[30] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[31] S. Wu and T. W. S. Chow, "Self-organizing and self-evolving neurons: A new neural network for optimization," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 385–396, Mar. 2007.

[32] S. Wu and T. W. S. Chow, "PRSOM: A new visualization method by hybridizing multi-dimensional scaling and self-organizing map," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1362–1380, Nov. 2005.

[33] H. Yin, "ViSOM: A novel method for multivariate data projection and structure visualization," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 237–243, Jan. 2002.

[34] I. Zelinka, "SOMA: Self-organizing migrating algorithm," in *New Optimization Techniques in Engineering*, G. C. Onwubolu and B. V. Babu, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 167–218.

[35] I. Zelinka, R. Senkerik, and E. Navratil, "Optimization of chaos control by means of evolutionary algorithms," in *Proc. 18th Int. Conf. Database Expert Syst. Applicat.*, Regensburg, Germany, Sep. 2007, pp. 163–167.

Lu Xu received the B.Eng. and M.S. degrees in electronic engineering from Beijing University of Technology, Beijing, China, in 2004 and 2007, respectively. She is currently working toward the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.

Her early interests were about robotics and artificial intelligence. Her current research interests include data clustering, optimization, and network routing.

Tommy Wai Shing Chow (M'94–SM'03) received the B.S. (First Hons.) and Ph.D. degrees from the University of Sunderland, Sunderland, U.K., in 1984 and 1988, respectively.

Since 1988, he has been with the City University of Hong Kong, Kowloon, Hong Kong, where he is currently a Professor with the Department of Electronic Engineering. He is an author and coauthor of numerous published works, including book chapters, and over 140 journal articles related to his research. His early research projects were mainly focused on supervised network problems. A number of fast training algorithms and weight initialization methods were developed. Recently he has focused on machine learning, neural network, and pattern recognition.

Dr. Chow received the Best Paper Award in the 2002 IEEE Industrial Electronics Society Annual Meeting, Seville, Spain. From 1997 to 1998, he was the Chairman of the Hong Kong Institute of Engineers, Control Automation and Instrumentation Division.