Self-Organizing and Self-Evolving Neurons: A New Neural Network for Optimization

Sitao Wu and Tommy W. S. Chow, *Senior Member, IEEE*

***Abstract*—A self-organizing and self-evolving agents (SOSENs) neural network is proposed. Each neuron of the SOSENs evolves itself with a simulated annealing (SA) algorithm. The self-evolving behavior of each neuron is a local improvement that results in speeding up the convergence. The chance of reaching the global optimum is increased because multiple SAs are run in a searching space. Optimum results obtained by the SOSENs are better in average than those obtained by a single SA. Experimental results show that the SOSENs have less temperature changes than the SA to reach the global minimum. Every neuron exhibits a self-organizing behavior, which is similar to those of the self-organizing map (SOM), particle swarm optimization (PSO), and self-organizing migrating algorithm (SOMA). At last, the computational time of parallel SOSENs can be less than the SA.**

***Index Terms*—Particle swarm optimization (PSO), self-organizing and self-evolving neurons (SOSENs), self-organizing map (SOM), simulated annealing (SA).**

I. Introduction

SIMULATED ANNEALING (SA) is a kind of stochastic optimization method introduced by Kirkpatrick *et al.* [1]. It is derived from the analogy of statistical mechanics and has been widely used in many large-scale optimization problems [2], [3]. However, it has a drawback of being computationally demanding. The ways of relieving its computational problem can be viewed from two directions. First, it is to parallelize the SA. Second, it is to hybridize the SA with other optimization algorithms like genetic algorithm (GA).

The operation of traditional SA algorithm is inherently serial and is difficult to be implemented in parallel because only a single solution evolves itself with time. This has made the speeding up of computational operation difficult. Since the last 20 years, efforts have been made for parallelizing the SA (PSA). The PSA can be mainly divided into four types: 1) data parallelism, 2) speculative algorithm, 3) multiple move, and 4) multiple independent runs (MIR) [4]. Data parallelism means that data is decomposed into small subsets distributed among processors [5]–[7]. Each processor handles its corresponding data subset and performs SA on it. However, data parallelism is problem-dependant and requires subsets to be loosely coupled. In speculative algorithms [8], steps of annealing are pipelined. Thus, subsequent decisions may affect the rejection of prior calculations. Multiple move parallelism allows evaluating multiple solutions belonging to a single Markov chain simultaneously on different processors [9], [10]. MIR parallelism computes several Markov chains, or parts of them, on different processors [4], [10]–[13]. The PSA can be implemented in parallel machines with single instruction multiple data, multiple instruction multiple data, shared memory, and distributed memory. Despite all these efforts, they have not radically changed the serial nature of the SA method [14].

In contrast to the SA, GA [15] uses a population of candidate solutions. The mechanism of GA is based on natural genetics and natural selection. As the entire candidate solutions can be easily distributed in a parallel way across processors, implementing GA in parallel is straightforward compared to the SA. However, there is no established convergence proof for the GA [16], whereas the property of convergence to global minima is theoretically proved for the SA method. The convergence of GA generally appears to be problem-dependent. Thus, many different methods were proposed. Also, the GA is not suitable for real-valued optimization problems because of the bit-string representations of real numbers.

The SA and GA can be hybridized into one new algorithm exhibiting the advantages of both the SA and GA. One of the hybrid algorithms is the genetic simulated annealing (GSA) proposed in [14], [17]–[21]. The population-based GSA algorithm hybridizes the GA and SA into one algorithm. The main difference between the GSA and GA is that the newly generated candidate solutions by the GSA are probabilistically accepted as child candidate solutions like the SA approach. This is different from the case of GA that uses roulette wheel selection. As the GSA still has the concept of parents and offspring, it is clearly more similar in nature to the GA than to the SA. Like the GA, the GSA needs to handle the problem of chromosome coding, which is not comfortable in handling real numbers. Apart from the GA, there are other population-based evolutionary optimization algorithms, such as genetic programming [22], evolution strategies [23], evolutionary programming [24], ant system [25], differential evolution (DE) [26], particle swarm optimization (PSO) [27], etc. Among these evolutionary algorithms, candidates cooperate and compete with each other during the course of optimization. The best solution is found in the final stage. All the candidate solutions can be considered as self-organized intelligent objects.

In this paper, a new neural network, called self-organizing and self-evolving neurons (SOSENs), is proposed. It is a population-based algorithm running a single SA in each neuron. As a

Manuscript received March 9, 2005; revised January 17, 2006; accepted September 10, 2006. This work was supported by the City University of Hong Kong under an SRG Grant 7001846-570.

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon SAR, Hong Kong (e-mail: 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.2006.887556

result, the optimization results can be enhanced compared with a single SA. The time of temperature changes by the SOSENs can also be reduced compared with that using the SA to reach the optimum value. The SOSENs are related to evolutionary algorithms because their self-organizing behavior is like the crossover operator in the GA. The self-evolving of the SOSENs is like the mutation operator in the GA. The SOSENs are also related to the MIR type PSA because their neurons run multiple Markov chains. The MIR-type PSA algorithms select the best solution, or mix the solutions according to the Boltzmann distribution for the synchronization at the next temperature [4], [10]–[13]. However, in the SOSENs, the synchronization mechanism is different in a way that new weights of all neurons are selected from the neighbors of the best neuron at the current temperature. In this paper, the SOSENs are studied from an evolutionary perspective and they are implemented in serial machines.

Like the GSA, the SOSENs can be considered as a combination of the SA and population-based evolutionary algorithms, but they have no concept of parents and offspring compared with the GSA. Each of their neuron learns by itself (self-evolving) with a SA, and learns from other neurons by cooperation and competition (self-organizing) after certain time. The behavior of self-evolving, or local improvement, has been adopted in memetic algorithm (MA) [28] and a hybrid genetic algorithm [29]. The self-organizing behavior of the SOSENs is similar to the SOM [30], [47], [50]. It is also similar to the PSO [27] and the self-organizing migrating algorithm (SOMA) [31], two recently developed population-based optimization algorithms. In the case of PSO, one parent particle generates one new child particle. For SOMA, one parent individual generates a series of new child individuals at the same time in a deterministic way and selects the best one. The self-organizing behavior is subsequently followed in the next step for the PSO and SOMA. Thus, both of them exhibit no self-evolving behavior. In the case of SOSENs, most of the original SA algorithm is not changed. The only change lies with the weights of neurons that are updated after the cooperation and competition caused by the self-organizing behavior, when simulated annealing in all candidates reaches the equilibrium at a certain temperature. Results obtained by the SOSENs can be better than running a single SA because input space is largely searched using multiple neurons. The time of temperature changes can be less than that of the SA to reach the optimum value. When each neuron of the SOSENs is distributed in processors of parallel machines, computation time is less than SA. Compared with other versatile evolutionary optimization algorithms, the SOSENs can deliver comparable performance or even better performance dependent on evaluation criteria and applications.

In this paper, Section II briefs the SA, and other algorithms. Section III describes the derivation, architecture, and implementation of the SOSENs. Section IV shows how the SOSENs optimize 16 synthetic continuous functions. In Sections V and VI, they are applied to the traveling salesman problem and fixed channel allocation (FCA) in mobile communications. Conclusion and discussion are given in Section VII.

II. Background

A. Simulated Annealing Algorithm

The SA is a flexible optimization algorithm originating in statistical mechanics [1]. It ensures that a new search will not be stuck in local minima. The mechanism of escaping from local minima is performed by a process to physical annealing. It allows "uphill moves" to higher cost at higher temperature to prevent optimization from being stuck in a local minimum. The general pseudocodes of the SA algorithm for searching global minimum are as follows:

1) Randomly initialize the solution $S = S_{\text{start}}$.

2) Set the initial temperature $T = T_0$.

3) Until equilibrium is reached, do:

   Generate a new solution S' from the neighborhood of S.

   Let E and E' be the values of the cost function at S and S', respectively.

   If $(E' < E)$, accept new solution $S = S'$.

   Else if $\exp(-(E' - E)/T) >$ a random number $\in [0, 1]$, accept new solution $S = S'$.

4) If stop criterion is valid, stop.

5) Reduce the temperature $T = \text{cooling}(T)$.

6) Go to 3).

B. Self-Organizing Map

The self-organizing map (SOM) [30], [48], [49] is well known with its ability to perform vector quantization while being able to preserve the topology of given data. It consists of N neurons located at a regular low dimensional [usually two-dimensional (2-D)] grid. The SOM algorithm is iterative. Each neuron i has a feature vector $w_i = [w_{i1}, \ldots, w_{id}]$ in a d-dimensional input space. At each training step t, a sample data vector $x(t)$ is randomly chosen from a training set. Distances between $x(t)$ and all feature vectors are computed. The winning neuron c is the neuron with the feature vector w_c closest to $x(t)$

$$c = \arg\min_i \|x(t) - w_i\|, \qquad i \in \{1, \ldots, N\} \tag{1}$$

where N is the number of neurons, and w_i is the feature vector of the ith neuron.

A set of neighboring neurons of the winning neuron c is denoted as N_c, which decreases its neighboring radius $\sigma(t)$ of the winning neuron with time. $h_{ic}(t)$ is defined as the neighborhood function around the winning neuron c at time t. It is a nonincreasing function with time and with the distance between neuron i and the winning neuron c. Usually, $h_{ic}(t)$ is chosen as a Gaussian function

$$h_{ic}(t) = \exp\left(-\frac{\|Pos_i - Pos_c\|^2}{2\sigma(t)^2}\right) \tag{2}$$

where Pos_i and Pos_c are the coordinates of the neurons i and c in the output grid, respectively.

The sequential weight-updating rule in the SOM algorithm can be written as

$$w_i(\mathrm{t}+1) = \begin{cases} w_i(t) + \varepsilon(t)h_{ic}(t)(x(t) - w_i(t)), & \forall i \in N_c \\ w_i(t), & \text{otherwise} \end{cases} \tag{3}$$

where $\varepsilon(t)$ is the learning rate decreasing with time. Note that (3) represents the self-organizing behavior, i.e., cooperation and competition.

C. PSO

The PSO [27] is a population-based optimization algorithm using multiple candidate solutions to find the global optimum of a search space. It is inspired mainly by social behavior of flock organisms, such as swarms of birds or schools of fishes. The population is called a swarm and an individual is called a particle. A particle moves with an adaptive speed with an attempt to find the global optimum through cooperating and competing with other particles. When a specific particle finds the best solution, other particles move closer to it.

Let the position vector and the velocity vector of particle i in the d-dimensional space be $X_i = [x_{i1}, \ldots, x_{id}]^T$ and $V_i = [v_{i1}, \ldots, v_{id}]^T$ respectively. Let $P_i = [p_{i1}, \ldots, p_{id}]^T$ be the best position previously encountered by particle i, and $P_g = [p_{g1}, \ldots, p_{gd}]^T$ be the best position of all particles. Then, the new velocity and the new position of particle i are modified [27], [32] by

$$v_{ij} = \omega v_{ij} + c_1 \,\mathrm{rand}\,(\cdot)(p_{ij} - x_{ij}) + c_2 \,\mathrm{rand}\,(\cdot)(p_{\mathrm{gj}} - x_{ij}), \quad j = 1, \ldots, d \tag{4}$$

$$x_{ij} = x_{ij} + v_{ij}, \qquad j = 1, \ldots, d \tag{5}$$

where ω is a positive parameter called inertia weight, c_1 and c_2 are positive constants called cognitive and social parameters respectively, and $\mathrm{rand}(\cdot)$ is a random number uniformly distributed in the range of [0, 1]. Note that (4) and (5) represent the self-organizing behavior, i.e., cooperation and competition.

D. DE

The DE [26] is also a population-based optimization algorithm. It uses floating-point encoding. Its parameter perturbation is self-adjusted by utilizing the information from populations themselves. To start with its self-adjusted population reproduction scheme, let the position vector of the ith population in the d-dimensional space be $X_i = [x_{i1}, \ldots, x_{id}]^T$. Then, the candidate vector $U_i = [u_{i1}, \ldots, u_{id}]$ of the ith population for the next generation is generated by (6), as shown at the bottom of the page, where r_1, r_2, and r_3 are three randomly selected population indices such that $r_1 \neq r_2 \neq r_3 \neq i$, $\mathrm{rand}_j[0, 1)$ is a random number $(\in [0, 1))$ selected for the jth dimension, k is a random dimension for the ith population, and $\mathrm{CR}(\in [0, 1.0])$ and $a(\in (0, 1.0])$ are real constants. Equation (6) ensures that u_{ij} is different from x_{ij} in at least one dimension. After mutation by (6), the next generation of x_{ij} is selected by

$$X_i = \begin{cases} U_i, & \text{if } f(U_i) \leqslant f(X_i) \\ X_i, & \text{otherwise} \end{cases} \tag{7}$$

where $f(x)$ is the target function to be minimized. Equations (6) and (7) represent the self-organizing behavior of cooperation and competition.

E. SOMA

The SOMA [31] is another population-based optimization method. Its mechanism is similar to that of the PSO. However, the local optimum, represented by P_i in the PSO, is not considered in the SOMA. Furthermore, not all the dimensions of an individual are modified during the course of self-organizing. The modified dimensions are randomly selected. Let the position vectors of individual k and the best individual g in the d-dimensional space be $X_k = [x_{k1}, \ldots, x_{kd}]^T$ and $X_g = [x_{g1}, \ldots, x_{gd}]^T$, respectively. First, a SOMA generates an integer vector V_k

$$V_{ki} = \begin{cases} = 1, \mathrm{rand}_i < C \\ = 0, \mathrm{rand}_i \geqslant C \end{cases}, \qquad i = 1, \ldots, d, \tag{8}$$

where rand_i is random number $(\in (0, 1))$ for the ith dimension, C is a predefined constant satisfying $0 < C < 1$, and V is used for the purpose of perturbation. The SOMA then generates a series of data points $Y(j)$ along the direction between individual k and the best one

$$Y_{ki}(j) = X_{ki} + V_{ki}D(j)(X_{gi} - X_{ki}) \tag{9}$$

where $D(j)$ is the jth movement defined as $D(j) = 0, \Delta, 2\Delta, \ldots, n\Delta, \Delta$ is the step size and $n\Delta$ is less than a maximum movement length Max_len. The data point with the best value among all $Y(j)'s$ is selected as the new candidate of individual k. Note that if the parameters ω and c_1 in (4) are set to zeros, the updating rule (5) of PSO is similar to the updating rule (9) of SOMA. In fact, (9) embodies the self-organizing behavior of cooperation and competition.

F. Self-Evolving

Both the PSO and SOMA algorithms exhibit the behavior of cooperation and competition, but new candidate solutions are not well fine-tuned to a state of near local optimum. This results in a long computational time or slow convergence. If each candidate solution is moved toward local optimum by self-evolving before cooperation and competition, the improvement is accumulative which results in speeding up the convergence. Self-evolving is adopted in [28] and [29] and is called local improvement. Let X_{local} be the best solution of an individual X and

$$u_{ij} = \begin{cases} x_{r_3 j} + a(x_{r_1 j} - x_{r_2 j}), & \text{if} \quad \mathrm{rand}_j[0, 1) \leqslant \mathrm{CR} \quad \text{or} \quad j = k \\ x_{ij}, & \text{otherwise} \end{cases}, \qquad j = 1, \ldots, d \tag{6}$$

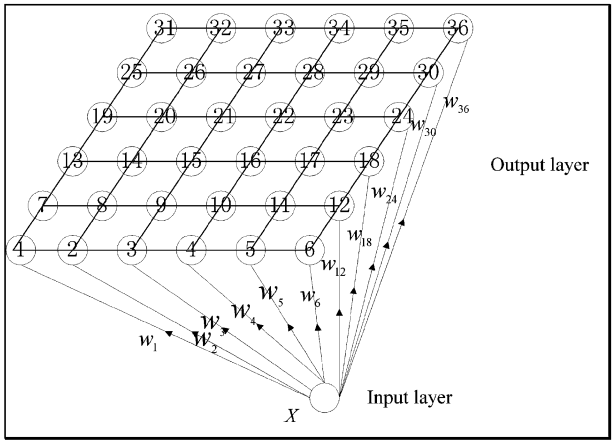

Fig. 1. Architecture of SOSENs.

$F(X)$ be an operator that changes X to one of its neighboring points. The pseudocode of self-evolving or local improvement is illustrated as follows.

Procedure of self-evolving or local improvement

Repeat

If $F(X)$ is better than X in local optimum

$X_{\text{best}} = F(X)$

End if

Until termination criterion for self-evolving or local improvement is satisfied.

III. SOSENs

The architecture of SOSENs is similar to the SOM. They also have input and output layers. Like the SOM, they consist of N output neurons located at a usually 2-D rectangular or hexagonal grid. An example of a rectangular SOSENs network with 6×6 neurons is shown in Fig. 1. Since the objective of SOSENs is to perform optimization, inputs and weights of neurons have different meanings from the traditional SOM. Inputs of SOSENs are the weight of a winner neuron representing the best solution at a time, while weights of neurons are the candidate solutions in population-based optimization algorithms. The final optimum solution is found among the weights of neurons in the SOSENs.

As discussed previously, the basic principle of the SOSENs includes two important properties. The mechanism of self-organizing has been used by some population-based optimization algorithms, such as GA, PSO, and SOMA; however, self-evolving is not adopted in these algorithms. On the other hand, a self-evolving mechanism has been adopted in MA [28], which is a GA hybridization with local search technique. If self-evolving is allowed in population-based optimization algorithms, optimization time will be reduced because self-evolving is a local improvement that fine-tunes the next candidate solution to the proximity of a local optimum, instead of randomly evolving. Given sufficient time for self-evolving, all neurons will be eventually self-organized. In fact, each neuron of SOSENs acts as an intelligent object that performs a single SA algorithm, cooperates, and competes with each other. Weights of neurons are updated toward the winner neuron that has the best optimization value at a time. Let W_i be the weight of the ith neuron of N neurons, i.e., the ith candidate solution of an optimization task. The detailed optimization procedure of SOSENs is as follows.

1) Randomly initialize the weight of the ith neuron $W_i (i = 1, \ldots, N)$, which satisfies certain optimization constraint.
2) Begin a SA process for each neuron at temperature $T = T_0$.
3) Under certain optimization constraint, each neuron evolves by a SA at temperature T.
4) When all SAs reach their equilibrium at temperature T, find the winner neuron c with the best optimum value among all the neurons.
   Update the weights $W_i's (i \neq c)$ of other neurons toward the winner neuron W_c according to the distances between them and the winner neuron on the fixed 2-D grid. The solution of the winner neuron c keeps unchanged. Under certain optimization constraint, the updating rule of all the neurons are as follows:

$$\begin{cases} W_i = W_i + \eta h_{ic}(W_c - W_i), & i \neq c \\ W_i = W_c, & i = c \end{cases} \tag{10}$$

   where η is the learning rate that is fixed over time and h_{ic} is the fixed neighborhood function defined by

$$h_{ic} = \exp\left(-\frac{\|a(Pos_i - Pos_c)\|^2}{2\sigma^2}\right) \tag{11}$$

   where Pos_i and Pos_c are the 2-D coordinates of neurons i and c in the 2-D grid, respectively, σ is the neighborhood radius, and a is a contraction coefficient.
5) If certain stop criterion is satisfied, stop. Otherwise, decrease the temperature T, go to 3).

The fixed learning rate η is in the range of [0, 1]. In this paper, η is set to 1 and satisfactory results are obtained for all the investigated problems. The neighborhood radius σ can be set to a range that affects all neurons in SOSENs. Large value of σ is used for easy optimization problem (smooth optimization surface) while small value of σ is used for hard optimization problem (complicated optimization surface). The number of neurons N can be set to smaller values when the optimization problem is easy. Larger value of N can be used when hard problem is to be optimized. The contraction coefficient a controls the extent of candidates being updated toward the winner neuron. A smaller value of a will generate a larger effect of pushing neurons toward the winner neuron. Usually, a can be chosen as 1. For very complicated optimization problems, a can be much less than 1 for escaping from a local minimum. Also, the distance between a neuron and the winner neuron affects the extent of that neuron being updated towards the winner in the solution space. That is, if a neuron is the one nearest (or farthest) to the winner candidate c, the extent of updating is the largest (or least) to the winner candidate in the solution space.

Self-evolving and self-organizing in a size 3×3 SOSENs is illustrated in Fig. 2 for solving a 2-D optimization problem. The initial positions of all the nine neurons are illustrated in Fig. 2(a).

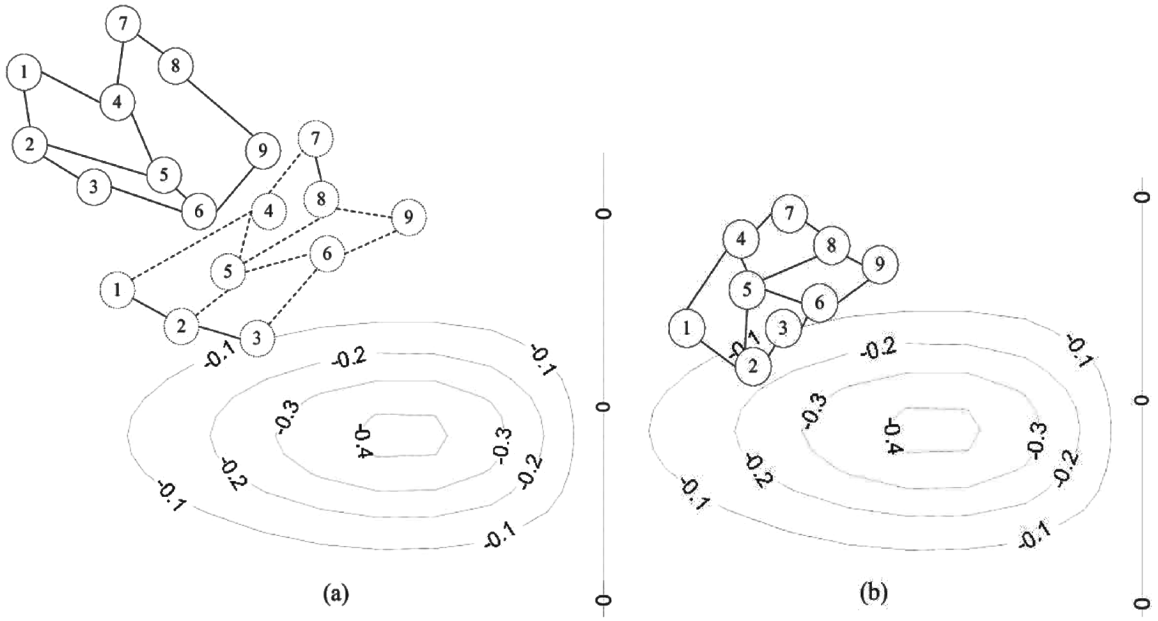

Fig. 2. Optimization process of SOSENs. (a) Solid circles are the initial positions of neurons and solid lines are the connections between neighboring neurons. Dotted circles are the positions of neurons and dotted lines are the connections between neighboring neurons when all SAs of SOSENs evolve and reach the equilibrium at temperature T. Note that neuron 3 is the winner candidate since its position is the nearest candidate to the global minimum (around -0.4). (b) The new positions of other neurons are self-organized around the winner neuron. The extent of updating is according to the distances between them and the winner neuron on the 2-D output grid.

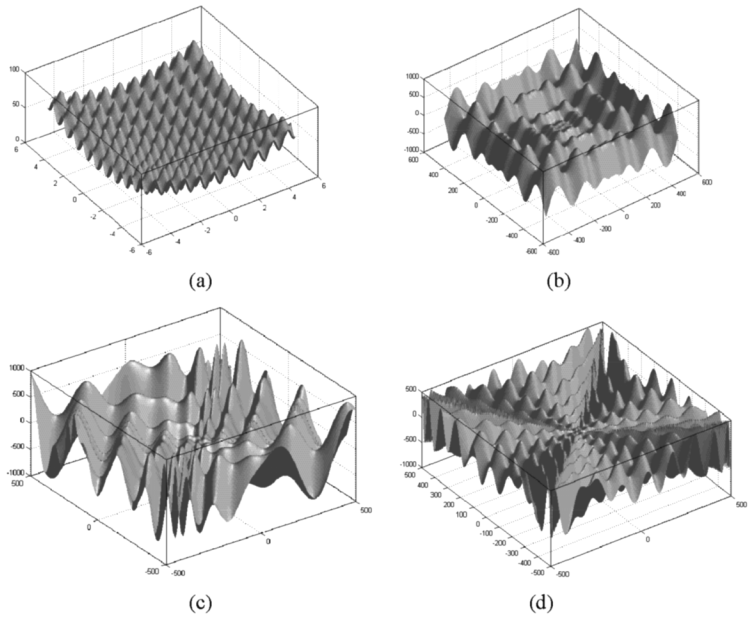

Fig. 3. Three-dimensional examples of (a) Rastrigin's function, (b) Schwefel's function, (c) Egg holder, and (d) Rana's function.

Then, all neurons evolve to their respective local optima by its SA, as shown in Fig. 2(a). After self-evolving, all the candidates are self-organized towards the neuron with the best optimum value at a time, as illustrated in Fig. 2(b).

IV. Experimental Results of SOSENs on 16 Synthetic Test Functions

In this section, 16 synthetic functions listed in Table I are tested [31]. The performances of the SOSENs on continuous variables are compared with the PSO, DE, SOMA, and SA. The SOSENs are implemented serially. That is, at a certain temperature, all the SAs in the SOSENs are performed in a way of one by one. The three-dimensional (3-D) examples of Rastrigin, Schwefel, Egg holder, and Rana's functions with two variables are shown in Fig. 3.

All the 16 test functions have n variables. In this paper, n is set to 100. Five optimization algorithms, the PSO, DE, SOMA, SA, and SOSENs (size 6×6), are applied to the optimization (minimization) problems. The common parameters of a single SA and SAs in the SOSENs are the same. The stopping criteria

TABLE I
SIXTEEN TEST FUNCTIONS

	Test function
1	First De Jong: $f(x)=\sum_{i=1}^{n}x_i^2,-5.12\le x_i\le 5.11$
2	Rosenbrock: $f(x)=\sum_{i=1}^{n-1}100(x_i^2-x_{i+1}^2)^2+(1-x_i^2)^2,-2.048\le x_i\le 2.047$
3	Third De Jong: $f(x)=\sum_{i=1}^{n}\lvert x_i\rvert,-2.048\le x_i\le 2.047$
4	Forth De Jong: $f(x)=\sum_{i=1}^{n}ix_i^4,-1.28\le x_i\le 1.27$
5	Rastrigin's function: $f(x)=20\sum_{i=1}^{n}\left(x_i^2-\cos(2\pi x_i)\right),\quad -5.12\le x_i\le 5.11$
6	Schwefel's function: $f(x)=-\sum_{i=1}^{n}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right),\quad -512.00\le x_i\le 511.00$
7	Griewangk's function: $f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\left(\frac{x_i}{\sqrt{i}}\right)+1,\quad -100.00\le x_i\le 100.00$
8	Stretched V sine wave function: $f(x)=\sum_{i=1}^{n-1}\left(x_i^2+x_{i+1}^2\right)^{0.25}\left(\sin\left(50\left(x_i^2+x_{i+1}^2\right)^{0.1}\right)^2+1\right),-10.00\le x_i\le 10.00$
9	Test function (Ackley): $f(x)=\sum_{i=1}^{n-1}e^{-0.2}\sqrt{x_i^2+x_{i+1}^2}+3(\cos(2x_i)+\cos(2x_{i+1})),\quad -30.00\le x_i\le 30.00$
10	Ackley's function: $f(x)=\sum_{i=1}^{n-1}20+e-20e^{-0.2\sqrt{0.5(x_i^2+x_{i+1}^2)}}-e^{0.5(\cos(2\pi x_i)+\cos(2\pi x_{i+1}))},\quad -30.00\le x_i\le 30.00$
11	Egg Holder: $f(x)=\sum_{i=1}^{n-1}-x_i\sin\left(\sqrt{\lvert x_i-(x_{i+1}+47)\rvert}\right)-(x_{i+1}+47)\sin\left(\sqrt{\left\lvert x_{i+1}+47+\frac{x_i}{2}\right\rvert}\right),-512.00\le x_i\le 512.00$
12	Rana's function: $f(x)=\sum_{i=1}^{n-1}\{x_i\sin(\sqrt{x_{i+1}+1-x_i})\cos(\sqrt{x_{i+1}+1+x_i})+(x_{i+1}+1)\sin(\sqrt{x_{i+1}+1+x_i})\cos(\sqrt{x_{i+1}+1-x_i})\},-500.00\le x_i\le 500.00$
13	Pathological function: $f(x)=\sum_{i=1}^{n-1}0.5+\frac{\sin^2\left(\sqrt{100x_i^2+x_{i+1}^2}\right)-0.5}{1+0.001\left(x_i^2-2x_ix_{i+1}+x_{i+1}^2\right)^2},\quad -100.00\le x_i\le 100.00$
14	Michalewicz's function: $f(x)=-\sum_{i=1}^{n-1}\left(\sin(x_i)\sin\left(\frac{x_i^2}{\pi}\right)^{20}+\sin(x_{i+1})\sin\left(\frac{2x_{i+1}^2}{\pi}\right)^{20}\right),\quad 0\le x_i\le\pi$
15	Inverted cosine wave function: $f(x)=-\sum_{i=1}^{n-1}e^{\frac{-\left(x_i^2+x_{i+1}^2+0.5x_ix_{i+1}\right)}{8}}\cos\left(4\sqrt{\left(x_i^2+x_{i+1}^2+0.5x_ix_{i+1}\right)}\right),-5.00\le x_i\le 5.00$
16	Sine envelope wave function: $f(x)=\sum_{i=1}^{n-1}\left(0.5+\frac{\sin\left(\sqrt{x_i^2+x_{i+1}^2}-0.5\right)^2}{1+0.001\left(x_i^2+x_{i+1}^2\right)^2}\right),-20.00\le x_i\le 20.00$

of both the SOSENs and SA are the difference between the current best value and the last best one being less than 10^{-5}. The neighborhood radius σ is set to 6. All the parameters of the SA and the SOSENs are listed in Table II. The parameters of the PSO, DE, and SOMA are selected by trials and errors such that they can produce good results. The numbers of population of PSO, DE, and SOMA for the first, third, and fourth functions are 20, while they are all 60 for the other functions.

For all the five optimization algorithms, they are repeated 100 times for each test function. The performance of the five algorithms with their corresponding minimum and mean cost values are listed in Table III. The numbers of temperature changes in the SA and SOSENs are also listed in Table III. All the five algorithms achieve similar performance. For the test function 2 (Rosenbrock), which is difficult to optimize [31], the SA and SOSENs exhibit much lower cost values than the other three algorithms. From the average results (mean value) listed in Table III, the PSO achieves the best results on two functions among the five algorithms. The DE, SOMA, SA, and SOSENs achieve the best on two, seven, two, and three functions, respectively. The SOMA may appear to be the best by rank, but we can perform comparison in another way. From the average results listed in Table III, if we choose the best result for each function among the five algorithms and compute the difference of each result to the best for each algorithm, as listed in Table IV, the average difference to the best for the SOSENs

TABLE II
PARAMETERS OF SA AND SOSENS FOR THE 16 TEST FUNCTIONS

Test function	The common parameters of a single SA and SAs in SOSENs ($T_0^* = 1.0$, $\alpha^{**} = 0.99$, $\varepsilon^{***} = 10^{-5}$)		The parameters only for SOSENs (size 6×6) ($\sigma = 6$)
	R****	L*****	a
1	5.0	10	1.0
2	0.01	1000	0.01
3	2.0	10	1.0
4	1.0	10	1.0
5	5.0	100	1.0
6	1500.0	100	1.0
7	10.0	100	1.0
8	10.0	4000	1.0
9	30.0	10	1.0
10	2.0	100	1.0
11	1000.0	10	1.0
12	500.0	10	1.0
13	100.0	100	1.0
14	2.0	100	1.0
15	5.0	50	1.0
16	20.0	50	1.0

*initial temperature, **temperature decrease by $T := \alpha T$, ***stopping criterion,
****neighborhood radius, *****number of iterations at a temperature,.

is the lowest on the 16 functions. From this perspective, the SOSENs deliver the best results.

The times of temperature change by the SOSENs is about $1/4$ in average of that by SA. This reduction is due to the fact that there are many candidates simultaneously searching the entire space while there is only one candidate used to search the space in the case of SA. Apparently, the SOSENs can find the global optimum more "efficiently" with less number of temperature changes compared with SA under the same stopping criterion. It is also worth noting that the times of temperature changes in SOSENs can be further reduced when more candidates are used. The actual central processing unit (CPU) time of serially implemented SOSENs is about eight times of that of the SA if the common parameters of SOSEN and SA are the same. It is difficult to compare the SOSENs to other three algorithms for actual CPU time because they are all parameter dependent. However, all the five methods can generate results in reasonable length of time. If the SOSENs are implemented in a parallel system, the computational time can be significantly less compared with a single SA.

V. EXPERIMENTAL RESULTS OF SOSENS ON TRAVELING SALESMAN PROBLEM

The SOSENs are used for solving the traveling salesman problem (TSP). In this case, SOSENs are used for discrete data. The TSP is a well-known optimization problem that is nondeterministic polynomial-time (NP)-hard. The TSP is a very important issue because it can be extended to other engineering problems such as vehicle routing, scheduling problem, printed circuit board design, etc. Among the methods used for solving the TSP, evolutionary algorithms appears to be very promising [33], [34]. In this paper, we consider the most-popular 2-D Euclidean TSP.

Assume that there is N cities, whose 2-D coordinates are expressed by $(x_i, y_i), i = 1, \ldots, N$. The cost function of TSP becomes

$$\sum_{i=1}^{N-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} + \sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}. \quad (12)$$

In order to find the local optimum for the TSP, some effective heuristic methods have been proposed. The three well-known methods are the 2-Opt [35], 3-Opt [36], and Lin–Kernighan (LK) algorithm [37]. The 2-Opt is the simplest one but the LK is much better than the other two. The neighborhood of a candidate generated by the LK has less cost value than the 2-Opt or 3-Opt. The LK has been considered the clear winner for local search algorithms [33]. Thus, it is used for generating neighborhood of a candidate by the SA or SAs in the SOSENs. Assume that there are N neurons for the SOSENs. The procedures of the SOSENs for solving the TSP are as follows.

1) Initialize each neuron $X^i, i = 1, \ldots, N$ which is an ordered list of cities.
2) Begin a SA process for each neuron at temperature $T = T_0$.

TABLE III
COMPARISON RESULTS FOR PSO, DE, SOMA, SA, AND SOSENS ON THE 16 TEST FUNCTIONS

Test function	Minimum/ Mean (Average number of temperature changes)				
	PSO	DE	SOMA	SA	SOSENs (6×6)
1	3.42×10^{-6}	1.40×10^{-1}	4.77×10^{-7}	3.60×10^{-2}	2.56×10^{-2}
	$/\,6.16\times10^{-4}$	$/\,5.46\times10^{-1}$	$/\,2.93\times10^{-3}$	$/\,3.30\times10^{-1}$ (1185.9)	$/\,1.16\times10^{-1}$ (246.8)
2	90.50	105.29	92.76	1.32×10^{-3}	1.77×10^{-3}
	/95.12	/148.10	/94.13	$/\,6.83\times10^{-3}$ (948.9)	$/\,1.97\times10^{-2}$ (826.0)
3	1.20×10^{-3}	2.63×10^{-6}	3.61×10^{-10}	3.39×10^{-1}	1.07×10^{-1}
	$/\,5.16\times10^{-3}$	$/\,3.57\times10^{-6}$	$/\,7.24\times10^{-10}$	$/\,3.78\times10^{-1}$ (2723.5)	$/\,3.92\times10^{-1}$ (381.5)
4	1.16×10^{-8}	3.72×10^{-9}	2.02×10^{-21}	9.75×10^{-2}	1.01×10^{-3}
	$/\,3.92\times10^{-8}$	$/\,1.00\times10^{-8}$	$/\,1.52\times10^{-20}$	$/\,1.52\times10^{-1}$ (566.4)	$/\,5.34\times10^{-3}$ (364.7)
5	-18766.25	-19952.61	-19701.51	-19999.50	-19999.84
	/-18364.29	/-18644.27	/-19588.09	/-19999.16 (2849.2)	/-19999.69 (415.5)
6	-29156.56	-41997.56	-41758.00	-41897.82	-41898.18
	/-27129.32	/-41819.80	/-41497.44	/-41895.79(1987.1)	/-41898.02 (381.4)
7	1.12×10^{-3}	2.01×10^{-5}	7.39×10^{-6}	7.39×10^{-2}	4.44×10^{-3}
	$/\,1.48\times10^{-2}$	$/\,7.51\times10^{-4}$	$/\,1.26\times10^{-5}$	$/\,4.18\times10^{-1}$ (835.6)	$/\,2.35\times10^{-2}$ (590.6)
8	3.40×10^{-1}	2.06×10^{-37}	2.16×10^{-2}	7.24×10^{-1}	7.00×10^{-1}
	/16.15	$/\,2.15\times10^{-3}$	$/\,2.58\times10^{-1}$	/1.79 (2342.7)	$/\,9.38\times10^{-1}$ (847.8)
9	-265.78	-290.60	-285.49	-287.76	-289.49
	/-218.32	/-288.25	/-282.36	/-286.16 (2979.0)	/-285.68 (407.1)
10	9.02×10^{-11}	-1.81×10^{-4}	-1.81×10^{-4}	1.59×10^{-1}	7.17×10^{-2}
	/335.64	$/\,-1.77\times10^{-4}$	$/\,-1.80\times10^{-4}$	/9.63 (2514.8)	$/\,1.49\times10^{-1}$ (446.8)
11	-50972.91	-330363.44	-74103.26	-67007.17	-68530.37
	/-41819.64	/-31856.50	/-69650.79	/-65506.04 (2934.1)	/-66067.52 (839.6)
12	-22761.68	-20228.18	-36174.74	-39918.04	-41047.18
	/-21763.26	/-19196.43	/-30693.02	/-38350.70(3038.2)	/-38005.11(988.80)
13	27.97	27.06	2.99	10.52	8.63
	/33.52	/28.63	/5.84	/15.05 (1551.7)	/13.11(625.8)
14	-95.30	-99.90	-99.70	-99.84	-99.89
	/-90.13	/-99.64	/-99.40	/-99.46 (751.5)	/-99.88 (451.7)
15	-99.00	-74.20	-88.41	-92.08	-95.76
	/-91.86	/-66.33	/-83.59	/-84.88 (843.5)	/-91.16 (493.10)
16	3.22×10^{-1}	13.17	3.98	4.55	5.06
	/4.81	/16.40	/4.47	/7.16 (1670.5)	/5.79 (713.8)

3) Each neuron evolves itself by SA at temperature T. The neighborhood of a neuron is generated by LK.
4) When all SAs reach their equilibrium, find the winner candidate c with the lowest value of the cost function (12).
5) Update the candidates $X^{i'}s(i \neq c)$ toward the winner candidate X^c according to the distances between them and the winner candidate on the 2-D output grid. The solution of the winner candidate c keeps unchanged. The solutions of direct neighboring candidates around the winner candidate c use LK once. The other nondirect neighboring candidates use multiple LKs, i.e., a chain of consecutive LKs. The farther the distance to the winner candidate c on the 2-D output grid, the more the number of multiple LKs is. If the distance between candidate i and the winner candidate c on the 2-D output grid is defined as D_{ic}, the number of multiple LKs is $\mathrm{ceil}(D_{ic})$, where $\mathrm{ceil}(x)$ is a function that

TABLE IV
DIFFERENCE TO THE BEST RESULTS FOR PSO, DE, SOMA, SA, AND SOSENS ON THE 16 TEST FUNCTIONS

Test function	Best result	Difference from the best results				
		PSO	DE	SOMA	SA	SOSENs (6×6)
1	6.16×10^{-4}	0.00	0.55	2.31×10^{-3}	0.33	0.12
2	6.83×10^{-3}	95.11	148.09	94.12	0.00	1.29×10^{-2}
3	7.24×10^{-10}	5.16×10^{-3}	3.57×10^{-6}	0.00	0.38	0.39
4	1.52×10^{-20}	3.92×10^{-8}	1.00×10^{-8}	0.00	0.15	5.34×10^{-3}
5	-19999.69	1635.4	1355.42	411.60	0.53	0.0
6	-41898.02	14768.70	78.22	400.58	2.23	0.0
7	1.26×10^{-5}	1.48×10^{-2}	7.38×10^{-4}	0.00	0.42	2.35×10^{-2}
8	2.15×10^{-3}	16.15	0.00	0.26	1.79	0.94
9	-288.25	69.93	0.00	5.89	2.09	2.57
10	-1.80×10^{-4}	335.64	3.00×10^{-6}	0.00	9.63	0.15
11	-69650.79	27831.15	37794.29	0.00	4144.75	3583.27
12	-38350.70	16587.44	19154.27	7657.68	0.0	345.59
13	5.84	27.68	22.79	0.0	9.21	7.27
14	-99.88	9.75	0.24	0.48	0.42	0.0
15	-91.86	0.00	25.53	8.27	6.98	0.70
16	4.47	0.34	11.93	0.00	2.69	1.32
Average		3836.08	3661.96	536.18	261.35	246.40

TABLE V
COMPARISON FOR HESEA, LKH, SA, AND SOSENS ON THE 11 TSP BENCHMARK PROBLEMS

Problem(Optimum value)	Average error (Average number of temperature changes)			
	HeSEA	LKH	SA	SOSENs (3×3)
lin318 (42029)	-*	0.027%	-(139.4)	-(24.9)
att532 (27686)	-	-	-(17.7)	-(3.1)
rat783 (8806)	-	-	-(3.0)	-(1.0)
pr1002 (259045)	-	-	-(47.2)	-(7.5)
vm1084 (239297)	-	0.007%	0.007% (1830.7)	-(559.6)
pcb1173 (56892)	-	-	-(415.0)	-(20.5)
u1432 (152970)	-	-	-(5.9)	-(1.1)
u2152 (64253)	-	0.001%	0.009% (1207.0)	-(274.6)
pr2392 (378032)	-	-	-(69.9)	-(12.7)
pcb3038 (137694)	-	-	-(461.0)	-(30.4)
fnl4461 (182566)	0.0005%	-	-(282.0)	-(131.6)

* "-" means that all the trials reach the optimum value.

takes the integer part of real number x. Thus, all the candidates are updated by $\mathrm{ceil}(D_{ic})$ LKs.

6) If certain stopping criterion is satisfied, stop. Otherwise, decrease the temperature T for all candidates, go to 3).

The single SA algorithm for the TSP is equivalent to the SOSENs network with only one neuron and without step 4). For SOSENs, a 3×3 network size is used for the 11 benchmark problems of Travelling Salesman LIBrary (TSPLIB) [39].

The optimum value of them is listed in Table V. The numbers of cities range from 318 to 4461. Four optimization algorithms, heterogeneous selection evolutionary algorithm (HeSEA) [34] (a genetic algorithm), Lin–Kernighan heuristic (LKH) [38] (an improved LK algorithm), SA, and SOSENs are compared. The performances of all these algorithms are evaluated by error defined by

$$\text{Error} = \frac{\text{average} - \text{optimum}}{\text{optimum}}. \tag{13}$$

The common parameters of the single SA and SAs in the SOSENs are the same. The number of iterations in a temperature for the SA and SOSENs is set to five for all the TSP problems. The stopping criterion for the SA and SOSENs is that the current best value reaches the optimum one, or the time of temperature changes reaches a predefined maximum number τ, which is set to five times of the number of cities for each problem.[1] The optimization task for each problem by all the four algorithms is repeated 20 times. All the results are listed in Table V. All the four algorithms achieve comparable performance on the 11 problems. For all the 11 problems, the optimum values are reached by the SOSENs in all 20 trials. However, for the "vm1084" and "u2152" problems, the SA cannot consistently reach to the optimum values in all 20 trials. The LKH cannot consistently reach to the optimum values for the "lin318," "vm1084," and "u2152" problems. The HeSEA cannot consistently reach to the optimum values for the "fnl4461" problem. Furthermore, the SOSENs need about $1/7$ of the times of temperature changes in average compared to the SA to reach the optimum values. In an extreme case of the "pcb1173" problem, the SOSENs need 20.5 average temperature changes while the SA needs 415.0 changes. If the SOSENs are implemented in parallel machines, there will be greatly computational time reduction.

VI. Experimental Results of SOSENs on FCA

In this section, we demonstrate the performance of the SOSENs using a real application of a FCA in wireless cellular systems. The SOSENs are also used for discrete data. In the universal mobile telecommunications system (UMTS) terrestrial radio access network (UTRAN), a geographical region is spatially divided into a number of cells. Research on how to efficiently utilize the scarce radio spectrum resource to satisfy the increasing users has become an exigent issue. Several types of channel assignment strategies have been proposed in different literatures. They can be classified into three types [40]: 1) FCA, 2) dynamic channel allocation (DCA), and 3) hybrid channel allocation (HCA). In FCA [41], a set of nominal channels is permanently assigned to each cell and there is no interference for these assigned channels. The predefined channel assignment is according to the estimated traffic load in each cell. If no unused nominal channels in a cell are available, a new call in that cell is blocked. It has been shown that the FCA is a generalized graph coloring problem [42] and is, therefore, NP-hard. Many methods have been proposed to provide near-optimal solutions. The FCA can be divided into four categories: 1) minimum order FCA, 2) minimum span FCA, 3) minimum interference FCA, and 4) minimum blocking FCA [43]. In this paper, minimum span FCA is used.

Minimum span FCA assigns channels so that no interferences occurs and tries to minimize frequency span. It can be expressed by

minimize: the difference of the highest frequency and the lowest frequency

s. t. demand constraint and no interference.

Assume that there are N cells for a cellular system. The demand constraint vector for all cells is represented by $D = [D_1, \ldots, D_N]^T$, where D_i is the current channel demand for the ith cell. A compatibility matrix C is introduced by considering the cochannel, adjacent and cosite interferences. The diagonal element C_{ii} of C indicates that any two channels assigned to cell i must be at least C_{ii} channels apart from each other to avoid cosite interference. The off-diagonal element $C_{ij}(\neq 0, i \neq j)$ means that any two channels assigned to cell i and j must be at least C_{ij} channels apart from each other to avoid cochannel or adjacent interferences. For the kth $(1 \leqslant k \leqslant D_i)$ call in the ith cell $(1 \leqslant i \leqslant N)$, a frequency f_{ik} is assigned. Note that f_{ik} is represented by an integer and the lowest frequency is represented by 0. Any different assigned frequencies should satisfy $|f_{ik} - f_{jl}| \geqslant C_{ij}$, where $i \neq j, k \neq l$.

Then, minimum span FCA can be formulated in by

$$\text{minimize}: \ \max_{i,k}(f_{ik}). \tag{14}$$

In the SOSENs, we use frequency-exhaustive strategy [44] to generate a neuron without violating the interference constraint. The weight of a neuron is an ordered call list. For generating a neighborhood of a neuron, we use the local search (LS) strategy in [44]. First, a call in the ith cell is randomly selected. Then, another call in the other cells is also randomly selected. Finally, the two selected frequencies are exchanged in the ordered call list and they determine the assignment by the frequency-exhaustive strategy.

The optimization procedures of the SOSENs for minimum-span FCA is like that for the TSP listed in Section V. One difference is that the weight of each neuron is ordered call list for the FCA. Another difference lies with the neighboring neurons that are updated by one or multiple LSs for the FCA. The last difference is that the cost function is (12) for FCA.

The single SA algorithm for the FCA is just the SOSENs with only one candidate and without step 4). The SOSENs and SA are performed on a Philadelphia, PA, benchmark cellular system in [45]. There are total of 21 cells in the system. There are total of eight problems with different interference constraints and demand vectors. The lower bounds of minimum spans for the eight problems have been extensively studied. All the configurations and lower bounds for the eight problems can be seen in [45]. The compatibility matrix C can be derived from the configurations. Simulations are conducted for 20 times by the SOSENs and SA. For the SOSENs network, a 2×2 size is used. The common parameters of the single SA and SAs in SOSENs are the same. The number of iteration at a fixed temperature for the SA and SOSENs is set to 500 for all the eight problems. The stopping criterion for them is that the current best value reaches the lower bound, or the time of temperature changes reaches a

[1]The results delivered by the LKH are generated by an executable file from http://www.akira.ruc.dk/~keld/research/lkh/. The results of the HeSEA are cited from [34].

TABLE VI
COMPARISON FOR PFGA, SA, AND SOSENS ON THE EIGHT PHILADELPHIA BENCHMARK PROBLEMS

Problem (Lower Bound)	Mean of minimum span (Average number of temperature changes)		
	PfGA	SA	SOSENs (2×2)
P1 (426)	426.00	426.00 (47.7)	426.00 (29.7)
P2 (426)	426.00	426.00 (48.3)	426.00 (29.0)
P3 (257)	259.38	260.1 (500.0)	258.9 (500.0)
P4 (252)	252.00	252.4 (324.0)	252.1 (260.6)
P5 (239)	239.00	239.7 (447.4)	239.3 (397.8)
P6 (179)	179.00	195.8 (500.0)	195.1 (500.0)
P7 (855)	855.00	855.00 (100.5)	855.00 (30.9)
P8 (524)	524.00	524.00 (134.2)	524.00 (104.1)

predefined maximum number τ, which is set to 500 for all the eight problems.

The comparative results of the SA, SOSENs, and parameter-free genetic algorithm (PFGA) [45] (a genetic algorithm) are listed in Table VI. They show that the SA and SOSENs can deliver comparable performance with the PFGA except problem 6. Except problem 3, the PFGA can reach the optimum values in all the trials. For problems 1, 2, 7, and 8, both the SOSENs and SA reach the lower bounds in all 20 trials. Compared to the SA, the SOSENs need about $1/2$ times of temperature changes in average to reach the lower bound. For problems 4 and 5, both the SOSENs and SA can reach the lower bounds in some of the 20 trials. However, the SOSENs achieve a little better average minimum span than the SA with about 85% times of temperature changes. For problem 3 and 6, both the SOSENs and SA cannot reach the lower bounds in all 20 trials within the predefined $\tau = 500$. However, the SOSENs achieve a little lower average minimum span than the SA. For problem 3, the average minimum span by the SOSENs is even lower the PfGA, but for problem 6, both the SA and SOSENs have a larger average minimum span than the PfGA. The average minimum span generated by the SA and SOSENs is about 16 more than the optimum value. Thus, both the SA and SOSENs are not good for problem 6. As multiple SAs are used to search the whole input space in the case of SOSENs, the chance of finding the global optimum is increased. Thus, SOSENs can have less temperature changes to reach the lower bound, or have lower frequency span in average than the SA. This means that when the SOSENs are implemented in parallel machines, the computational time can be largely reduced.

VII. CONCLUSION AND DISCUSSION

The SA is a serial optimization algorithm while the SOSENs are a population-based optimization algorithm using multiple SAs with self-evolving and self-organizing. When SOSENs have only one candidate, they have no self-organizing behavior and are thus equivalent to the SA. As multiple SAs are used to search the whole input space in the case of SOSENs, the optimization results are better in average than the SA for the cost values or the number of temperature changes. Although SOSENs use multiple SAs and thus are computationally heavier than a single SA as a whole, they require much less number of temperature changes for completing the optimization. As a result, when they are implemented by parallel machines, the optimization time can be largely reduced compared with a single SA. Furthermore, compared with the complex and tricky parallel implementation of serial SA, the proposed SOSENs can be implemented directly or easily in a parallel system because they are naturally a population-based optimization algorithm.

Although the mechanism of SOSENs is based on SA, tabu search algorithm [46] may be an alternative option to be utilized in each candidate. In this case, each candidate runs the tabu search algorithm with self-evolving and self-organizing. The tabu search, however, is a local search algorithm that cannot guarantee global optimum convergence. Also, the tabu search algorithm needs a tabu list to memorize the recently found solutions in order to search for the local optimum. Apparently, tabu-based SOSENs approach will be rather computationally demanding. Thus, tabu search algorithm is not recommended in the SOSENs.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers' useful suggestions that greatly improved the format of this paper.

REFERENCES

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[2] S. A. Kravitz and R. A. Rutenbar, "Placement by simulated annealing on a multiprocessor," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 6, no. 4, pp. 534–549, Jul. 1987.
[3] M. Duque-Antón, D. Kunz, and B. Rüber, "Channel assignment for cellular radio using simulated annealing," *IEEE Trans. Veh. Technol.*, vol. 42, no. 1, pp. 14–21, Feb. 1993.
[4] A. Bevilacqua, "A methodological approach to parallel simulated annealing on an SMP system," *J. Parallel Distrib. Comput.*, vol. 62, pp. 1548–1570, 2002.
[5] J. R. A. Allwright and D. B. Carpenter, "A distributed implementation of simulated annealing for the traveling sales man problem," *Parallel Comput.*, vol. 10, pp. 335–338, 1989.
[6] P. Banerjee, M. Jones, and J. Sargent, "Parallel simulated annealing algorithms for cell placement on hypercube multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 1, no. 1, pp. 91–106, Jan. 1990.
[7] A. Casotto, F. Romeo, and A. Sangiovanni-Vincentelli, "A parallel simulated annealing algorithm for the place of macro-cell," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 6, no. 5, pp. 838–847, May 1987.
[8] Sohn, "Parallel N-ary speculative computation of simulated annealing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 6, no. 10, pp. 997–1005, Oct. 1995.

[9] P. R. Ragot and G. Dreyfus, "A problem independent parallel implementation of simulated annealing: Models and experiments," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 9, no. 8, pp. 827–835, Aug. 1990.
[10] R. Diekmann, R. Lüling, and J. Simon, "Problem independent distributed simulated annealing and its applications," in *Lecture Notes in Economics and Mathematical Systems*. Berlin, Germany: Springer-Verlag, 1993, vol. 396, pp. 17–44.
[11] S. Gupta and L. Bic, "Distributed adaptive simulated annealing for synthesis design space exploration" Univ. California, Irvine, CA, Tech. Rep. 99-05, 1999.
[12] S. M. Bhandarkar, S. Machaka, S. Chirravuri, and J. Arlond, "Parallel Computing for chromosome reconstruction via ordering of DNA sequences," *Parallel Comput.*, vol. 24, no. 12–13, pp. 1177–1204, 1998.
[13] K.-W. Chu, Y. Deng, and J. Reinitz, "Parallel simulated annealing by mixing of states," *J. Comput. Phys.*, vol. 148, pp. 646–662, 1999.
[14] H. Chen, N. S. Flann, and D. W. Watson, "Parallel genetic simulated annealing: A massively parallel SIMD algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 2, pp. 126–136, Feb. 1998.
[15] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1994.
[16] G. Rudolf, "Convergence properties of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96–101, Jan. 1994.
[17] K. Jeong and J. J. Lee, "Adaptive simulated annealing genetic algorithm for system identification," *Eng. Applicat. Artif. Intell.*, vol. 9, no. 5, pp. 523–532, 1996.
[18] H. C. Huang, J. S. Pan, Z. M. Lu, S. H. Sun, and H. M. Hang, "Vector quantization based on genetic simulated annealing," *Signal Process.*, vol. 81, pp. 1513–1523, 2001.
[19] M. Kolonko, "Some new results on simulated annealing applied to job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 113, pp. 123–136, 1999.
[20] P. Wong and S. Y. W. Wong, "Hybrid genetic/simulated annealing approach to short-term multiple-fuel-constrained generation scheduling," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 776–784, May 1997.
[21] M. E. Aydin and T. C. Fogarty, "A distributed evolutionary simulated annealing algorithm for combinational optimization problems," *J. Heuristics*, vol. 10, pp. 269–292, 2004.
[22] R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
[23] H.-G. Beyer, *The Theory of Evolution Strategies*. Berlin, Germany: Springer-Verlag, 2001.
[24] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1996.
[25] 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.
[26] J. Lampinen and R. Storn, "Differential evolution," in *New Optimization Techniques in Engineering*, G. Onwubolu and B. V. Babu, Eds. Heidelberg, Germany: Springer-Verlag, 2004, pp. 123–166.
[27] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, Nov. 1995, pp. 1942–1948.
[28] P. Moscato, C. Cotta, and A. Mendes, "Memetic algorithms," in *New Optimization Techniques in Engineering*, G. Onwubolu and B. V. Babu, Eds. Heidelberg, Germany: Springer-Verlag, 2004, pp. 53–85.
[29] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1424–1437, Nov. 2004.
[30] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer-Verlag, 1997.
[31] Zelinka, "SOMA-self-organizing migrating algorithm," in *New Optimization Techniques in Engineering*, G. Onwubolu and B. V. Babu, Eds. Heidelberg, Germany: Springer-Verlag, 2004, pp. 167–217.
[32] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. IEEE Int. Congr. Evol. Comput.*, 1999, vol. 3, pp. 101–106.
[33] S. Jung and B.-R. Moon, "Toward minimal restriction of genetic encoding and crossovers for the two-dimensional euclidean TSP," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 557–565, Dec. 2002.
[34] H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao, "An evolutionary algorithm for large traveling salesman problems," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 34, no. 4, pp. 1718–1829, Aug. 2004.
[35] H. Braun, "On traveling salesman problems by genetic algorithm," in *Proc. Workshop Parallel Problem Solving from Nature*, 1990, pp. 129–133.
[36] P. Jog, J. Suh, and D. Gucht, "The effect of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 110–115.
[37] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Oper. Res.*, vol. 21, no. 4598, pp. 498–516, 1973.
[38] Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, pp. 106–130, 2000.
[39] G. Reinelt, "TSPLIB-A traveling salesman library," *ORSA J. Comput.*, vol. 3, pp. 376–384, 1991.
[40] Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," *IEEE Personal Commun.*, vol. 3, no. 3, pp. 10–30, Jun. 1996.
[41] W. C. Y. Lee, *Mobile Cellular Telecommunications Systems*. New York: McGraw-Hill, 1989.
[42] H. Tamura, M. Sengoku, S. Shinoda, and T. Abe, "Channel assignment problem in a cellular system and a new coloring problem of networks," *IEICE Trans. Commun. Electron. Inf. Syst.*, vol. 74, no. 10, pp. 2983–2989, 1991.
[43] H. G. Sandalidis and P. Stavroulakis, "Heuristics for solving fixed-channel assignment problems," in *Handbook of Wireless Networks and Mobile Computing*, Stojmenovic, Ed. New York: Wiley, 2002, pp. 51–70.
[44] W. Wang and C. K. Rushforth, "An adaptive local-search algorithm for the channel-assignment problem (CAP)," *IEEE Trans. Veh. Technol.*, vol. 45, no. 3, pp. 459–466, Aug. 1995.
[45] S. Matsui, I. Watanabe, and K.-I. Tokoro, "Application of the parameter-free genetic algorithm to the fixed channel assignment problem," *Syst. Comput. Jpn.*, vol. 36, no. 4, pp. 350–359, 2005.
[46] F. Glover and M. Laguna, *Tabu Search*. Boston, MA: Kluwer, 1997.
[47] 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.
[48] E. Berglund and J. Sitte, "The parameterless self-organizing map algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 305–316, Mar. 2006.
[49] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 586–600, May 2000.
[50] T. W. S. Chow and S. Wu, "An online cellular probabilistic self-organizing map for static and dynamical data sets," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 51, no. 4, pp. 732–747, Apr. 2004.

Sitao Wu received the B.E. and M.E. degrees from the Department of Electrical Engineering, Southwest Jiaotong University, Chengdu, P.R. China, in 1996 and 1999, respectively, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, P.R. China, in 2004.

His research interest areas are neural networks, pattern recognition, and their applications.

Tommy W. S. Chow (M'94–SM'03) received the B.Sc. (First Honors) degree and the Ph.D. degree from the Department of Electrical and Electronic Engineering, University of Sunderland, Sunderland, U.K., in 1984 and 1988, respectively, working on a collaborative project between The International Research and Development, Newcastle Upon Tyne, U.K. and the Ministry of Defense (Navy) U.K. He undertook his Trainee with Reyrolle Technology, U.K.

He is a Professor in the Department of Electronic Engineering at the City University of Hong Kong, Hong Kong, P.R. China. He has been working on different consultancy projects with the Mass Transit Railway, Kowloon-Canton Railway Corporation, Hong Kong. He has also conducted other collaborative projects with the Kong Electric Co. Ltd, and Royal Observatory Hong Kong, and the MTR Hong Kong on the application of neural networks for machine fault detection and forecasting. He is an author and coauthor of numerous published works, including book chapters, and over 100 journal articles related to his research. His main research has been in the area of learning theory and optimizations, system identification, and machine fault diagnostics.

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