A Multiagent Evolutionary Algorithm for Combinatorial Optimization Problems

Jing Liu, *Member, IEEE*, Weicai Zhong, *Member, IEEE*, and Licheng Jiao, *Senior Member, IEEE*

***Abstract*—Based on our previous works, multiagent systems and evolutionary algorithms (EAs) are integrated to form a new algorithm for combinatorial optimization problems (CmOPs), namely, MultiAgent EA for CmOPs (MAEA-CmOPs). In MAEA-CmOPs, all agents live in a latticelike environment, with each agent fixed on a lattice point. To increase energies, all agents compete with their neighbors, and they can also increase their own energies by making use of domain knowledge. Theoretical analyses show that MAEA-CmOPs converge to global optimum solutions. Since deceptive problems are the most difficult CmOPs for EAs, in the experiments, various deceptive problems with strong linkage, weak linkage, and overlapping linkage, and more difficult ones, namely, hierarchical problems with treelike structures, are used to validate the performance of MAEA-CmOPs. The results show that MAEA-CmOP outperforms the other algorithms and has a fast convergence rate. MAEA-CmOP is also used to solve large-scale deceptive and hierarchical problems with thousands of dimensions, and the experimental results show that MAEA-CmOP obtains a good performance and has a low computational cost, which the time complexity increases in a polynomial basis with the problem size.**

***Index Terms*—Combinatorial optimization problems (CmOPs), deceptive problems, evolutionary algorithms (EAs), hierarchical problems, multiagent systems.**

NOTATION LIST

$\boldsymbol{\mathcal{S}}$	Search space.
$\boldsymbol{E}$	Set of all the different energy values.
E^i	ith element of $\boldsymbol{E}$.
$\boldsymbol{\mathcal{S}}^i$	Set of elements in $\boldsymbol{\mathcal{S}}$ whose energy is equal to E^i.
$\boldsymbol{x}$, $\boldsymbol{a}$, and $\boldsymbol{c}$	Binary vectors in $\boldsymbol{\mathcal{S}}$.
$\boldsymbol{x}^*$	Best binary vector in $\boldsymbol{\mathcal{S}}$.
$\boldsymbol{L}$	Agent lattice.
$\boldsymbol{L}_{i,j}$	Agent located at the ith row and jth column of $\boldsymbol{L}$.
$\boldsymbol{L}^t$	Agent lattice in the tth generation.
$\boldsymbol{\mathcal{L}}$	Set of all agent lattices.
$\boldsymbol{\mathcal{L}}^i$	Set of agent lattices whose energy is equal to E^i.
$\boldsymbol{L}^{ij}$	jth agent lattice in $\boldsymbol{\mathcal{L}}^i$.
$\boldsymbol{N}_{i,j}$	Set of neighbors of $\boldsymbol{L}_{i,j}$.
$\boldsymbol{T}$	Main learning table.
$\boldsymbol{T}^q$	qth sublearning table.
n	Dimension of $\boldsymbol{\mathcal{S}}$.
a_i, c_i, and l_i	ith components of $\boldsymbol{a}$, $\boldsymbol{c}$, and $\boldsymbol{L}_{i,j}$, respectively.
L_{size}	Size of the agent lattice.
r	Perception range.
$T_{i,j}$	Positive integer located at the ith row and the jth column in $\boldsymbol{T}$.
$T^q_{i,j}$	Positive integer located at the ith row and the jth column in $\boldsymbol{T}^q$.
s	Number of sublearning tables.
$p_{ij.kl}$	Probability of transition from $\boldsymbol{L}^{ij}$ to $\boldsymbol{L}^{kl}$.
$p_{ij.k}$	Probability of transition from $\boldsymbol{L}^{ij}$ to any agent lattice in $\boldsymbol{\mathcal{L}}^k$.
$p_{i.k}$	Probability of transition from any agent lattice in $\boldsymbol{\mathcal{L}}^i$ to any agent lattice in $\boldsymbol{\mathcal{L}}^k$.
$f(\boldsymbol{x})$	Objective function.
u	Number of variables whose value is 1 in $f(\boldsymbol{x})$.
f^{mapping}	Mapping function in hierarchical problems.
$U(0,1)$	Uniform random real number in [0, 1].
$Energy(\bullet)$	Energy of an agent.
$Learning(\bullet)$	Flag to determine which strategy is used in the self-learning behavior.
$\vert \bullet \vert$	Cardinality of a set.
$\Pr\{\bullet\}$	Probability of the event in "{}."

I. INTRODUCTION

EVOLUTIONARY algorithms (EAs) [1]–[6] are stochastic global optimization methods inspired by the biological mechanisms of evolution and heredity. In recent years, with the characteristics of easier application, greater robustness, and better parallel processing than most classical optimizing methods, EAs have been widely used for numerical optimization, combinatorial optimization, classification, and many other engineering problems [7]–[13]. But it is realized from practice that EAs still have weakness, and it is worth stepping back and exploring how to best learn from nature and how to incorporate our existing knowledge in artificial intelligence into EAs.

Agent-based computation has been studied for several years in the field of distributed artificial intelligence [14], [15] and has been widely used in other branches of computer science [7], [8], [16]–[18]. Multiagent systems are computational systems in which several agents interact or work together to achieve some purposes. Problem solving is an area with which many multiagent-based applications are concerned. It includes distributed solutions to problems, solving distributed problems,

Manuscript received October 21, 2008; revised February 15, 2009. First published July 28, 2009; current version published October 30, 2009. This work was supported by the National Natural Science Foundation of China under Grant 60872135 and by the Program for New Century Excellent Talents of the University of China under Grant NCET-06-0857. This paper was recommended by Associate Editor H. Takagi.

The authors are with the Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China (e-mail: neouma@163.com; wakenov@gmail.com; lchjiao@mail.xidian.edu.cn).

Digital Object Identifier 10.1109/TSMCB.2009.2025775

and distributed techniques for problem solving [14], [15]. Many researches have been done in this field. Liu *et al.* [17] introduced an application of distributed techniques for solving constraint satisfaction problems (CSPs). They solved 7000-queen problems by an energy-based multiagent model.

On the other hand, there are two related previous works we have done. First, multiagent systems and genetic algorithms (GAs) are integrated to solve global numerical optimization problems in [7], and the proposed method can find high-quality solutions at a low computational cost even for functions with 10 000 dimensions. Second, multiagent systems and EAs are combined to form a new algorithm for solving CSPs in [8], and the comparison results show that the proposed method outperforms several famous existing algorithms. All these results show that both agents and EAs have high potentials in solving complex and ill-defined problems.

Following our pervious works, multiagent systems and EAs are integrated to solve combinatorial optimization problems (CmOPs) in this paper. CmOPs are one of the most basic and important research and application fields. Usually, they are nondifferentiable, discontinuous, multidimensional, constrained, and highly nonlinear NP-hard problems and have lots of local optima. With the intrinsic properties of CmOPs in mind, we design two agent behaviors, that is, the competition behavior and the self-learning behavior, to realize the purpose of minimizing the objective function values. Based on this, a new algorithm, namely, MultiAgent EA for CmOPs (MAEA-CmOPs), is proposed. Theoretical analyses show that MAEA-CmOPs converge to global optimum solutions.

In the experiments, since deceptive problems are the most difficult CmOPs for EAs, deceptive problems with various linkages and more difficult ones, namely, hierarchical problems with treelike structures, are used to validate the performance of MAEA-CmOPs. The slow convergence rate is one of the key reasons that prevent EAs from practical applications. However, the experimental results show that MAEA-CmOP has a fast convergence rate and obtains a good performance even for various deceptive and hierarchical problems with thousands of dimensions. These results demonstrate that MAEA-CmOP is a competent algorithm for practical applications.

Compared with our previous works [7], [8], the common point between MAEA-CmOPs and the works in [7] and [8] is that they all follow the idea of integrating multiagent systems into EAs. However, since they cope with different problems, the meaning of agents is different, so the designed agent behaviors are completely different. Agent behaviors are the core of each algorithm, which decides that the three algorithms are totally different in both implementation and application fields. Apart from this, the experiments in each work are performed on the famous benchmark problems in each field.

Since MAEA-CmOP uses a lattice-based population, it is similar to cellular GAs (CGAs) [19]–[22] to some extent. However, all operations of CGAs are the same with those of traditional GAs except that CGAs have a neighborhood structure. In essence, CGAs are greedy techniques for enabling a fine-grained parallel implementation of GAs and can present the same problem of premature convergence of traditional GAs [22]. However, MAEA-CmOP makes use of the ability of agents in sensing and acting on the environment and puts emphasis on designing behaviors for agents. The experimental results show that MAEA-CmOP achieves a good performance even for deceptive and hierarchical problems with thousands of dimensions, which demonstrate that MAEA-CmOP overcomes the problem of premature convergence to some extent.

The rest of this paper is organized as follows: Section II describes the agents for CmOPs. Section III describes the implementation of MAEA-CmOPs and analyzes its convergence. Sections IV and V present experimental studies on the various deceptive and hierarchical problems, respectively. Finally, Section VI concludes the works in this paper.

II. Agents for CmOPs

According to [15] and [17], an agent is a physical or virtual entity that essentially has the following properties: 1) it is able to live and act in an environment; 2) it is able to sense its local environment; 3) it is driven by certain purposes; and 4) it has some reactive behaviors. In general, four elements should be defined when multiagent systems are used to solve problems. The first is the meaning and purpose of each agent. The second is the environment where all agents live. Since each agent has only local perception, the third is the definition of the local environment. The last is the behavior that each agent can take to achieve its purposes. In what follows, these elements for CmOPs are defined.

A. Definition of Agents

The objective of CmOPs is to optimize some functions to satisfy the given constraints over a discrete and finite mathematical structure. It can be described as follows: Given a problem $(\boldsymbol{\mathcal{S}}, f)$, where $\boldsymbol{\mathcal{S}}$ is the search space, and f is the objective function, the objective is to find $\boldsymbol{x}^* \in \boldsymbol{\mathcal{S}}$, which satisfies $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$ for $\forall\, \boldsymbol{x} \in \boldsymbol{\mathcal{S}}$. Since the search space is discrete, each element in it can be encoded by a binary string. Based on this, an agent is defined as follows.

Definition 1: An *agent*, labeled as $\boldsymbol{a}$, represents a candidate solution for the CmOP under consideration and is encoded by a binary vector

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \boldsymbol{\mathcal{S}}, \qquad a_i = 0 \text{ or } 1; \quad 1 \leq i \leq n \quad (1)$$

where n is the problem size. The energy of $\boldsymbol{a}$ is equal to its associated objective function value, namely, $Energy(\boldsymbol{a}) = f(\boldsymbol{a})$. The purpose of $\boldsymbol{a}$ is to increase its energy as much as possible.

All agents live in a toroidal latticelike environment, which is called as agent lattice and labeled as $\boldsymbol{L}$. The size of $\boldsymbol{L}$ is $L_{\text{size}} \times L_{\text{size}}$, where L_{size} is a positive integer. Each agent is fixed on a lattice point and can only interact with its neighbors. Therefore, the agent lattice can be represented as the form in Fig. 1. Supposing that the agent located at (i, j) is represented as $\boldsymbol{L}_{i,j},\ i, j = 1, 2, \ldots, L_{\text{size}}$, then the set of neighbors of $\boldsymbol{L}_{i,j}$, labeled as $\boldsymbol{N}_{i,j}$, is determined by a parameter, namely, *perception range* (r), as

$$N_{i,j} = L_{k,l}$$

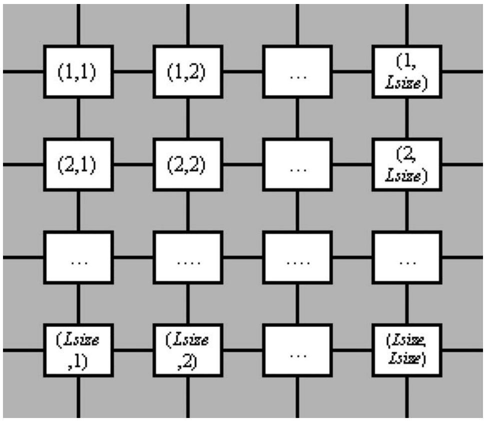

Fig. 1. Model of the agent lattice, where each cell denotes an agent, and the numbers in it are the row and column positions.

where

$$\begin{cases} (i-r) \le k \le (i+r) \text{ and } k = \begin{cases} k + L_{\text{size}}, & k < 1 \\ k - L_{\text{size}}, & k > L_{\text{size}} \end{cases} \\ (j-r) \le l \le (j+r) \text{ and } l = \begin{cases} l + L_{\text{size}}, & l < 1 \\ l - L_{\text{size}}, & l > L_{\text{size}} \end{cases} \end{cases} . \tag{2}$$

B. Behaviors of Agents

For CmOPs, the purpose of an algorithm is to find out the best solutions incurring a computational cost as low as possible. Thus, the computational cost can be considered as the resources of the environment in which all agents live. Each agent will compete with others to gain more resources. At the same time, each agent can also increase its energy by using its knowledge. Based on this, two agent behaviors, namely, the competition behavior and the self-learning behavior, are designed. Since all agents live in a lattice environment, each agent can only interact with its neighbors. To let each behavior be more flexible, the parameter *perception range* is used to adjust the neighbors in each behavior.

1) Competition Behavior: The perception range in this behavior is fixed to 1. Thus, each agent has eight neighbors. For $\boldsymbol{L}_{i,j}$, its energy is compared with their neighbors' energies. If $L_{i,j}$'s energy is the maximum, then $\boldsymbol{L}_{i,j}$ can survive; otherwise, $\boldsymbol{L}_{i,j}$'s lattice point will be occupied by the child of the agent whose energy is the maximum in $\boldsymbol{N}_{i,j}$. The details are described as follows.

Suppose this behavior is performed by the agent located at (i,j). Let $\boldsymbol{L}_{i,j} = (l_1, l_2, \ldots, l_n)$ and $\boldsymbol{a}_{\max} = (a_1, a_2, \ldots, a_n)$ be the agents with the maximum energy in $\boldsymbol{N}_{i,j}$. If $Energy(\boldsymbol{L}_{i,j}) < Energy(\boldsymbol{a}_{\max})$, then a child agent $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ is generated from $\boldsymbol{a}_{\max}$ to replace $\boldsymbol{L}_{i,j}$. There are two strategies to generate $\boldsymbol{c}$:

Strategy 1:

$$c_i = \begin{cases} a_i, & U(0,1) < 0.5 \\ l_i, & \text{otherwise} \end{cases}, \qquad 1 \le i \le n \tag{3}$$

where $U(0,1)$ is a uniform random real number in [0, 1].

Strategy 2:

$$c_i = \begin{cases} a_i, & U(0,1) > 1/n \\ 1 - a_i, & \text{otherwise} \end{cases}, \qquad 1 \le i \le n. \tag{4}$$

Strategy 1 generates a child agent $\boldsymbol{c}$ by making use of both information in $\boldsymbol{L}_{i,j}$ and $\boldsymbol{a}_{\max}$, whereas Strategy 2 is a kind of bit mutation that is commonly used in EAs. Since both $\boldsymbol{L}_{i,j}$ and $\boldsymbol{a}_{\max}$ are binary vectors, their differences can be measured by the Hamming distance. Clearly, the smaller the Hamming distance between $\boldsymbol{L}_{i,j}$ and $\boldsymbol{a}_{\max}$, the more similar these two agents are, and then the lower the probability of generating a better agent with Strategy 1. Thus, we use the following rule to determine which strategy is selected: If the ratio of the Hamming distance between $\boldsymbol{L}_{i,j}$ and $\boldsymbol{a}_{\max}$ to n is larger than 0.5, then Strategy 1 is selected; otherwise, Strategy 2 is selected.

2) Self-Learning Behavior: The purpose of this behavior is to increase the energy of an agent as much as possible. However, the resources in the environment are limited. As a result, an agent can obtain a self-learning opportunity only when its energy is larger than those of their neighbors. First, a learning table is defined as follows.

Definition 2: A *Learning Table*, labeled as $(\boldsymbol{T})_{p\times 2}$, is a matrix with p rows and two columns. Let $T_{i,j}$ be the positive integer located at the ith row and the jth column. Then, a learning table must satisfy the following conditions:

$$(1 \le T_{i,j} \le n) \text{ and } (T_{i,1} \le T_{i,2}), \qquad 1 \le i \le p; \quad j = 1 \text{ or } 2 \tag{5}$$

$$\forall\, i \ne j, \qquad (T_{i,1} \ne T_{j,1}) \text{ or } (T_{i,2} \ne T_{j,2}) \tag{6}$$

$$p \le \frac{n(n+1)}{2} \tag{7}$$

where $(\boldsymbol{T})_{(n(n+1)/2)\times 2}$ is the main learning table, and any set of rows of $(\boldsymbol{T})_{(n(n+1)/2)\times 2}$ is a sublearning table.

The memory needed to store the main learning table is determined by n. In what follows, we consider the case with $n < 2^{16}$. If we use 2 B to store each data. and there are $n(n+1)$ data in total, then the total bytes needed to store a main learning table are $2n(n+1)$, namely, $(n(n+1)/2^{19})$ megabytes. When $n = 1000$, 1.9 MB is needed, whereas for $n = 5000$, 47.7 MB is needed. It is clear that when the problem size is small, we can directly store the whole main learning table; but when the problem size is large, it is difficult. Thus, the main learning table is divided into s sublearning tables, which are labeled as $\boldsymbol{T}^1, \boldsymbol{T}^2, \ldots, \boldsymbol{T}^s$, and each one has $(n(n+1)/2s)$ rows, so that they are fit for the available memory.

Suppose $\boldsymbol{L}_{i,j} = (l_1, l_2, \ldots, l_n)$ obtains a self-learning opportunity. Then, two self-learning strategies can be used, which are given in Algorithms 1 and 2, respectively, where $Learning(\boldsymbol{L}_{i,j})$ is a Boolean flag attached to each agent to determine which strategy is used.

Algorithm 1 Self-Learning Strategy 1

Step 1: Let $q \leftarrow 1$.

Step 2: Generate $\boldsymbol{T}^q$.

Step 3: Randomly select a row j from $\boldsymbol{T}^q$; generate $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ according to

$$a_i = \begin{cases} l_i, & \left(i < T^q_{j,1}\right) \text{ or } \left(i > T^q_{j,2}\right) \\ 1 - l_i, & \text{otherwise} \end{cases}, \qquad 1 \leq i \leq n. \tag{8}$$

Step 4: If $Energy(\boldsymbol{a}) > Energy(\boldsymbol{L}_{i,j})$, then let $Learning(\boldsymbol{a}) \leftarrow False$, $\boldsymbol{L}_{i,j} \leftarrow \boldsymbol{a}$, and stop.

Step 5: Delete the jth row from $\boldsymbol{T}^q$; if $\boldsymbol{T}^q$ is empty, then let $q \leftarrow q + 1$.

Step 6: If $q \leq s$, then go to Step 2; otherwise, $Learning(\boldsymbol{L}_{i,j}) \leftarrow True$, and stop.

Algorithm 2 Self-Learning Strategy 2

Step 1: Generate a permutation of $1, 2, \ldots, n$, that is, $(p_1, p_2, \ldots, p_n)$; $q \leftarrow 1$.

Step 2: Generate $\boldsymbol{T}^q$.

Step 3: Randomly select a row j from $\boldsymbol{T}^q$; generate $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ according to

$$a_{p_i} = \begin{cases} l_{p_i}, & \left(i < T^q_{j,1}\right) \text{ or } \left(i > T^q_{j,2}\right) \\ 1 - l_{p_i}, & \text{otherwise} \end{cases}, \qquad 1 \leq i \leq n. \tag{9}$$

Step 4: If $Energy(\boldsymbol{a}) > Energy(\boldsymbol{L}_{i,j})$, then let $Learning(\boldsymbol{a}) \leftarrow False$, $\boldsymbol{L}_{i,j} \leftarrow \boldsymbol{a}$, and stop.

Step 5: Delete the jth row from $\boldsymbol{T}^q$; if $\boldsymbol{T}^q$ is empty, then let $q \leftarrow q + 1$.

Step 6: If $q \leq s$, then go to Step 2; otherwise, stop.

The first strategy iteratively selects a segment of $\boldsymbol{L}_{i,j}$ and reverses it until the energy of $\boldsymbol{L}_{i,j}$ is increased or the main learning table is empty. The second strategy first rearranges $\boldsymbol{L}_{i,j}$ and then iteratively selects a segment of $\boldsymbol{L}_{i,j}$ and reverses it until the energy of $\boldsymbol{L}_{i,j}$ is increased. When the energy of $\boldsymbol{L}_{i,j}$ cannot be increased by the first strategy, the probability of increasing the energy by the same strategy in the future is very low. Thus, $Learning(\boldsymbol{L}_{i,j})$ is set to true when the first strategy fails to increase the energy. That is to say, usually, the first strategy is used, and only when $Learning(\boldsymbol{L}_{i,j})$ is true is the second strategy used instead.

III. MAEA-CmOPs and Its Convergence

A. Implementation of MAEA-CmOPs

To solve CmOPs, all agents must orderly adopt the competition behavior and the self-learning behavior. Here, the two behaviors are controlled by means of evolution so that the agent lattice can evolve generation by generation. At each generation, the competitive behavior is first performed by each agent. As a result, the agents with low energy are cleaned out from the agent lattice so that there is more space developed for the agents with higher energy. Then, the self-learning behavior is performed by some good agents. This process is performed iteratively until the stop criteria are satisfied. The details are given in Algorithm 3.

Algorithm 3 MAEA-CmOPs

Step 1: Initialize the agent lattice $\boldsymbol{L}^0$: generate $(L_{\text{size}} \times L_{\text{size}})$ agents, and let $Learning(\boldsymbol{L}_{i,j}) \leftarrow False$, where $i, j = 1, 2, \ldots, L_{\text{size}}$; $t \leftarrow 0$.

Step 2: If the termination criteria are satisfied, then output the agent with maximum energy in the current agent lattice and stop.

Step 3: Perform the competition behavior on each agent in $\boldsymbol{L}^t$, that is, if $\forall\, \boldsymbol{a} \in \boldsymbol{N}_{i,j}$, $Energy(\boldsymbol{a}) \leq Energy(\boldsymbol{L}^t_{i,j})$, then let $\boldsymbol{L}^{t+(1/2)}_{i,j} \leftarrow \boldsymbol{L}^t_{i,j}$; otherwise, select a strategy to generate a new agent $\boldsymbol{c}$, $Learning(\boldsymbol{c}) \leftarrow False$, and $\boldsymbol{L}^{t+(1/2)}_{i,j} \leftarrow \boldsymbol{c}$.

Step 4: Perform the self-learning behavior on each agent in $\boldsymbol{L}^{t+(1/2)}$, that is, if $\forall\, \boldsymbol{a} \in \boldsymbol{N}_{i,j}, Energy(\boldsymbol{a}) \leq Energy(\boldsymbol{L}^{t+(1/2)}_{i,j})$ and $Learning(\boldsymbol{L}^{t+(1/2)}_{i,j}) = False$, then perform Algorithm 1 on $\boldsymbol{L}^{t+(1/2)}_{i,j}$; if $\forall\, \boldsymbol{a} \in \boldsymbol{N}_{i,j}$, $Energy(\boldsymbol{a}) \leq Energy(\boldsymbol{L}^{t+(1/2)}_{i,j})$ and $Learning(\boldsymbol{L}^{t+(1/2)}_{i,j}) = True$, then perform Algorithm 2 on $\boldsymbol{L}^{t+(1/2)}_{i,j}$; let $\boldsymbol{L}^{t+1}_{i,j} \leftarrow \boldsymbol{L}^{t+(1/2)}_{i,j}$.

Step 5: Let $t \leftarrow t + 1$, and go to Step 2.

In traditional EAs, individuals that can generate offspring are usually selected from the whole population. Thus, the global fitness distribution of the population must be determined in advance. However, in nature, a global selection does not exist, and the global fitness distribution cannot be determined either. In fact, the real natural selection only occurs in a local environment, and each individual can only interact with those around it. That is, in some phases, the natural evolution is just a kind of local phenomenon. The information can be shared globally only after a process of diffusion.

Algorithm 3 shows that, in MAEA-CmOPs, since each agent can only sense its local environment, its behaviors can only take place between it and its neighbors. There is no global selection at all, so the global fitness distribution is not required. An agent interacts with its neighbors so that information can be transferred to them. In such a manner, the information is diffused to the whole agent lattice. As can be seen, the evolutionary mechanism based on the agent lattice used in MAEA-CmOPs is closer to the real evolutionary mechanism in nature than that based on the population model used in traditional EAs.

B. Convergence of MAEA-CmOPs

The search space $\boldsymbol{\mathcal{S}}$ is a discrete state space, so the number of elements of $\boldsymbol{\mathcal{S}}$ is finite. Thus, the number of all different energy values is finite since each element can only have one energy value. Let the set of all different energy values be $\boldsymbol{E}$, namely

$$\boldsymbol{E} = \{Energy(\boldsymbol{a}) \mid \boldsymbol{a} \in \boldsymbol{\mathcal{S}}\} = \left\{E^1, E^2, \ldots, E^{|\boldsymbol{E}|}\right\} \tag{10}$$

where $E^1 > E^2 > \cdots > E^{|\boldsymbol{E}|}$. Clearly, E^1 is the global optimum solution. This immediately gives us the opportunity to partition $\boldsymbol{\mathcal{S}}$ into a collection of nonempty subsets, namely, $\{\boldsymbol{\mathcal{S}}^i\}$, where

$$\boldsymbol{\mathcal{S}}^i = \left\{\boldsymbol{a} \mid \boldsymbol{a} \in \boldsymbol{\mathcal{S}} \text{ and } Energy(\boldsymbol{a}) = E^i\right\}, \quad i = 1, 2, \ldots, |\boldsymbol{E}|. \tag{11}$$

$\boldsymbol{\mathcal{S}}^1$ consists of all agents whose energies are E^1.

Let the energy of an agent lattice $\boldsymbol{L}$ be labeled as $Energy(\boldsymbol{L})$, which is equal to the energy of the best agent in $\boldsymbol{L}$. Let $\boldsymbol{\mathcal{L}}$ be the set of all agent lattices. Thus, $\boldsymbol{\mathcal{L}}$ can be partitioned into a collection of nonempty subsets, namely, $\{\boldsymbol{\mathcal{L}}^i\}$, where

$$\boldsymbol{\mathcal{L}}^i = \left\{ \boldsymbol{L} \mid \boldsymbol{L} \in \boldsymbol{\mathcal{L}} \text{ and } Energy(\boldsymbol{L}) = E^i \right\}, \quad i = 1, 2, \ldots, |\boldsymbol{E}|. \quad (12)$$

$\boldsymbol{\mathcal{L}}^1$ consists of all agent lattices whose energies are E^1.

Let $\boldsymbol{L}^{ij}$, $i = 1, 2, \ldots, |\boldsymbol{E}|$, $j = 1, 2, \ldots, |\boldsymbol{\mathcal{L}}^i|$, be the jth agent lattice in $\boldsymbol{\mathcal{L}}^i$. During the evolutionary process, $\boldsymbol{L}^{ij}$ is transformed into another one, namely, $\boldsymbol{L}^{kl}$, and this process can be viewed as a transition from $\boldsymbol{L}^{ij}$ to $\boldsymbol{L}^{kl}$. Let $p_{ij.kl}$ be the probability of transition from $\boldsymbol{L}^{ij}$ to $\boldsymbol{L}^{kl}$, $p_{ij.k}$ be the probability of transition from $\boldsymbol{L}^{ij}$ to any agent lattice in $\boldsymbol{\mathcal{L}}^k$, and $p_{i.k}$ be the probability of transition from any agent lattice in $\boldsymbol{\mathcal{L}}^i$ to any agent lattice in $\boldsymbol{\mathcal{L}}^k$. Then, we have the following theorem for MAEA-CmOPs.

Theorem 1: In MAEA-CmOPs, $\forall \boldsymbol{L}^{ij} \in \boldsymbol{\mathcal{L}}^i$, $i = 1, 2, \ldots, |\boldsymbol{E}|$, $j = 1, 2, \ldots, |\boldsymbol{\mathcal{L}}^i|$, we have 1) $\forall k > i$, $p_{i.k} = 0$, and 2) $\exists k < i, p_{i.k} > 0$.

Proof: Letting $\boldsymbol{L}^{ij}$ be the agent lattice in the tth generation, which is labeled as $\boldsymbol{L}^t$ for convenience, and letting $\boldsymbol{a}^t$ be the agent with maximum energy in $\boldsymbol{L}^t$, then we have $Energy(\boldsymbol{a}^t) = E^i$.

1) According to Step 3 of Algorithm 3, we have $\boldsymbol{a}^t \in \boldsymbol{L}^{t+(1/2)}$. Because Step 4 of Algorithm 3 can only increase the energy of agents, we have

$$Energy(\boldsymbol{L}^{t+1}) \geq Energy(\boldsymbol{L}^t) \Rightarrow \forall k > i, p_{ij.kl} = 0$$
$$\Rightarrow \forall k > i, p_{ij.k} = \sum_{l=1}^{|\mathcal{L}^k|} p_{ij.kl} = 0 \Rightarrow \forall k > i, p_{i.k} = 0. \quad (13)$$

2) Letting $\boldsymbol{a}^{t+(1/2)}$ be the agent with the maximum energy in $\boldsymbol{L}^{t+(1/2)}$, then we have $Energy(\boldsymbol{a}^{t+(1/2)}) \geq Energy(\boldsymbol{a}^t)$. Thus, there are two cases:

Case 2.1) $Energy(\boldsymbol{a}^{t+(1/2)}) > Energy(\boldsymbol{a}^t)$. It is clear that $\exists k < i, p_{i.k} > 0$.

Case 2.2) $Energy(\boldsymbol{a}^{t+(1/2)}) = Energy(\boldsymbol{a}^t)$. According to Step 4 of Algorithm 3, $\boldsymbol{a}^{t+(1/2)}$ will obtain a self-learning opportunity. Supposing $\exists \boldsymbol{a}'$, $Energy(\boldsymbol{a}') = E^k > E^i$, without loss of generality, the values of the (i_1)th, (i_2)th, $\ldots$, $(i_{n'})$th bits in $\boldsymbol{a}'$ are different from those corresponding bits in $\boldsymbol{a}^{t+(1/2)}$ and $(i_1 < i_2 < \cdots < i_{n'})$.

According to the values of $Learning(\boldsymbol{a}^{t+(1/2)})$ and $(i_1, i_2, \ldots, i_{n'})$, there are three cases to determine the probability of transition from $\boldsymbol{a}^{t+(1/2)}$ to $\boldsymbol{a}'$, which is labeled as $\Pr\{\boldsymbol{a}^{t+(1/2)} \to \boldsymbol{a}'\}$:

1) $Learning(\boldsymbol{a}^{t+(1/2)}) = True$: The self-learning behavior is performed by $\boldsymbol{a}^{t+(1/2)}$ with Algorithm 2. According to Step 1 of Algorithm 2, there are $(n!)$ permutations of n integers, and only $((n - n' + 1)! \times n'!)$ permutations can make $i_1, i_2, \ldots, i_{n'}$ succeed to each other. Based on the definition of learning table and Steps 2–6 of Algorithm 2, we have

$$\Pr\{\boldsymbol{a}^{t+1/2} \to \boldsymbol{a}'\} > \left(\frac{1}{\frac{n(n+1)}{2}} \times \frac{(n - n' + 1)! \times n'!}{n!} \right) > 0 \quad (14)$$

where $\Pr\{\bullet\}$ denotes the probability of the event in "{}." Therefore, $\exists k < i, p_{i.k} > 0$.

2) $Learning(\boldsymbol{a}^{t+(1/2)}) = False$ and $(\forall 1 \leq j < n', i_{j+1} - i_j = 1)$: The self-learning behavior is performed by $\boldsymbol{a}^{t+(1/2)}$ with Algorithm 1. Based on the definition of learning table and Steps 2–6, we have

$$\Pr\{\boldsymbol{a}^{t+1/2} \to \boldsymbol{a}'\} > \left(1 \middle/ \frac{n(n+1)}{2} \right) > 0. \quad (15)$$

Therefore, $\exists k < i, p_{i.k} > 0$.

3) $Learning(\boldsymbol{a}^{t+(1/2)}) = False$ and $(\exists 1 \leq j < n', i_{j+1} - i_j > 1)$: The self-learning behavior is performed by $\boldsymbol{a}^{t+(1/2)}$ with Algorithm 1. Clearly, any row of the main learning table cannot transform $\boldsymbol{a}^{t+(1/2)}$ to $\boldsymbol{a}'$. If Algorithm 1 stops at Step 4, then it demonstrates that a better agent has been found; otherwise, $Learning(\boldsymbol{a}^{t+(1/2)})$ is set to True, and $\boldsymbol{a}^{t+(1/2)}$ is added into $\boldsymbol{L}^{t+1}$. Apparently, we have $\boldsymbol{a}^{(t+1)+(1/2)} \geq \boldsymbol{a}^{t+1} \geq \boldsymbol{a}^{t+(1/2)}$. If $\boldsymbol{a}^{(t+1)+(1/2)} > \boldsymbol{a}^{t+(1/2)}$, then it demonstrates that $\boldsymbol{L}^t$ has been transformed into the agent lattice with higher energy; otherwise, if $\boldsymbol{a}^{(t+1)+(1/2)} = \boldsymbol{a}^{t+(1/2)}$, $\boldsymbol{a}^{t+(1/2)}$ can obtain a self-learning opportunity. At this moment, $Learning(\boldsymbol{a}^{t+(1/2)}) = True$. Therefore, $\exists k < i, p_{i.k} > 0$. ■

This theorem shows that there is always a positive probability to transit from an agent lattice to another with higher energy and a zero probability to another with lower energy. Thus, once MAEA-CmOP enters $\boldsymbol{\mathcal{L}}^1$, it will never go out. Before proving the convergence of MAEA-CmOPs, we first revisit an important existing theorem.

1) Theorem 2 [23]: Let $\boldsymbol{P}' : n \times n$ be a reducible stochastic matrix, which means that by applying the same permutations to rows and columns, $\boldsymbol{P}'$ can be brought into the form $\begin{pmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{T} \end{pmatrix}$, where $\boldsymbol{C} : m \times m$ is a primitive stochastic matrix, and $\boldsymbol{R}, \boldsymbol{T} \neq 0$. Then

$$\begin{aligned} \boldsymbol{P}'^{\infty} &= \lim_{k \to \infty} \boldsymbol{P}'^k \\ &= \lim_{k \to \infty} \begin{pmatrix} \boldsymbol{C}^k & \boldsymbol{0} \\ \sum_{i=0}^{k-1} \boldsymbol{T}^i \boldsymbol{R} \boldsymbol{C}^{k-i} & \boldsymbol{T}^k \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{C}^{\infty} & \boldsymbol{0} \\ \boldsymbol{R}^{\infty} & \boldsymbol{0} \end{pmatrix} \end{aligned} \quad (16)$$

is a stable stochastic matrix with $\boldsymbol{P}'^{\infty} = \boldsymbol{1}'\boldsymbol{p}'^{\infty}$, where $\boldsymbol{p}'^{\infty} = \boldsymbol{p}'^{0}\boldsymbol{P}'^{\infty}$ is unique regardless of the initial distribution, and $\boldsymbol{p}'^{\infty}$ satisfies $p_i'^{\infty} > 0$ for $1 \leq i \leq m$ and $p_i'^{\infty} = 0$ for $m < i \leq n$.

Based on Theorems 1 and 2 and [24], the convergence of MAEA-CmOPs is proved as follows.

Theorem 3: MAEA-CmOP converges to the global optimum solutions.

Proof: It is clear that one can consider each $\boldsymbol{\mathcal{L}}^i$, $i = 1, 2, \ldots, |\boldsymbol{E}|$, as a state in a homogeneous finite Markov chain. According to Theorem 1(1), the transition matrix $\boldsymbol{P}'$ of the Markov chain can be written as follows:

$$\boldsymbol{P}' = \begin{pmatrix} p_{1.1} & 0 & \cdots & 0 \\ p_{2.1} & p_{2.2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_{|\boldsymbol{E}|.1} & p_{|\boldsymbol{E}|.2} & \cdots & p_{|\boldsymbol{E}|.|\boldsymbol{E}|} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{T} \end{pmatrix} \quad (17)$$

where $\boldsymbol{C} = (p_{1.1})$, $\boldsymbol{R} = (p_{2.1}, p_{3.1}, \ldots, p_{|\boldsymbol{E}|.1})^T$, and $\boldsymbol{T} = \begin{pmatrix} p_{2.2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ p_{|\boldsymbol{E}|.2} & \cdots & p_{|\boldsymbol{E}|.|\boldsymbol{E}|} \end{pmatrix}$. Theorem 1 (2) shows that $\boldsymbol{R} \neq \boldsymbol{0}$, $\boldsymbol{T} \neq \boldsymbol{0}$, and $\boldsymbol{C} = (p_{1.1}) = (1)$ is a primitive stochastic matrix. Thus, $\boldsymbol{P}'$ is a reducible stochastic matrix and satisfies the conditions in Theorem 2. Therefore, $\boldsymbol{P}'^{\infty}$ is a stable stochastic matrix and is equal to

$$\begin{aligned} \boldsymbol{P}'^{\infty} &= \lim_{k\to\infty} \boldsymbol{P}'^{k} \\ &= \lim_{k\to\infty} \begin{pmatrix} \boldsymbol{C}^k & \boldsymbol{0} \\ \sum_{i=0}^{k-1} \boldsymbol{T}^i \boldsymbol{R} \boldsymbol{C}^{k-i} & \boldsymbol{T}^k \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{C}^{\infty} & \boldsymbol{0} \\ \boldsymbol{R}^{\infty} & \boldsymbol{0} \end{pmatrix}. \end{aligned} \quad (18)$$

Since $\boldsymbol{P}'^{\infty}$ is a stochastic matrix, the summation of any row in $\boldsymbol{P}'^{\infty}$ must be equal to 1. Then, we have $\boldsymbol{C}^{\infty} = (1)$, and $\boldsymbol{R}^{\infty} = (1, 1, \ldots, 1)^T$, that is,

$$\boldsymbol{P}'^{\infty} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}. \quad (19)$$

Therefore

$$\lim_{t\to\infty} \Pr\left\{ Energy(\boldsymbol{L}^t) = E^1 \right\} = 1. \quad (20)$$

This implies that MAEA-CmOP converges to the global optimum solutions. ■

IV. Experiments on Deceptive Problems

Goldberg *et al.* considered in [25] that deceptive problems are important test functions for testing GAs or other algorithms with similar search mechanisms. Therefore, we use various large-scale deceptive functions, which are constructed by four small-scale deceptive functions, namely, subfunctions, in common use to test the performance of MAEA-CmOPs in this section. These four subfunctions are given in (21)–(24), where the value of each variable is set to 0 or 1, and u represents the number of variables whose value is 1.

Goldberg's three-order deceptive function

$$f_{\mathrm{Goldberg3}}(a_1, a_2, a_3) = \begin{cases} 30, & (a_1 = 1) \text{ and } (a_2 = 1) \text{ and } (a_3 = 1) \\ 28, & (a_1 = 0) \text{ and } (a_2 = 0) \text{ and } (a_3 = 0) \\ 26, & (a_1 = 0) \text{ and } (a_2 = 0) \text{ and } (a_3 = 1) \\ 22, & (a_1 = 0) \text{ and } (a_2 = 1) \text{ and } (a_3 = 0) \\ 14, & (a_1 = 1) \text{ and } (a_2 = 0) \text{ and } (a_3 = 0) \\ 0, & \text{otherwise} \end{cases}. \quad (21)$$

Three-order deceptive function [26]

$$f_{\mathrm{deceptive3}}(a_1, a_2, a_3) = \begin{cases} 0.9, & u = 0 \\ 0.8, & u = 1 \\ 0, & u = 2 \\ 1, & u = 3 \end{cases}. \quad (22)$$

Five-order trap function [27]

$$f_{\mathrm{trap5}}(a_1, a_2, a_3, a_4, a_5) = \begin{cases} 5, & u = 5 \\ 4 - u, & \text{otherwise} \end{cases}. \quad (23)$$

Six-order bipole deceptive function [26]

$$f_{\mathrm{bipolar6}}(a_1, a_2, a_3, a_4, a_5, a_6) = \begin{cases} 0.9, & u = 3 \\ 0.8, & (u = 2) \text{ and } (u = 4) \\ 0, & (u = 1) \text{ and } (u = 5) \\ 1, & (u = 0) \text{ and } (u = 6) \end{cases}. \quad (24)$$

Apparently, the global optimum solutions of $f_{\mathrm{Goldberg3}}$, $f_{\mathrm{deceptive3}}$, and f_{trap5} are the vectors with all values equal to 1, and those of f_{bipolar6} are the vectors with all values equal to 1 or 0. The four subfunctions above have different complexity and properties, so the functions made up of them can validate an algorithm's performance comprehensively.

According to the properties of variables in subfunctions, the deceptive functions can be divided into three classes. The first class is strong-linkage deceptive functions whose variables are connected to each other. The second class is weak-linkage deceptive functions whose variables are not connected to each other. The sets of variables in different subfunctions of both of these two kinds of functions are not intersected. Thus, the third class is overlapping-linkage functions whose sets of variables in different subfunctions are intersected. In this section, all these three kinds of deceptive functions are used to test the performance of MAEA-CmOPs.

Some parameters must be assigned before MAEA-CmOPs can be used to solve problems. First, since $L_{\mathrm{size}} \times L_{\mathrm{size}}$ is equivalent to the population size in traditional EAs, L_{size} can be selected from 3 to 10 in general and is set to 5 here. Second, because s is used to adjust the size of the learning table so that it can be fit for the memory available, and because it has no effect on the performance, it is set to 1 here. Third, since r of the competition behavior is fixed to 1, it does not need to be adjusted. Fourth, r of the self-learning behavior is used to control how many agents can obtain the self-learning opportunity. The smaller it is, the more agents can obtain the self-learning opportunity, then the higher the computational

TABLE I
AVERAGE NUMBER OF FUNCTION EVALUATIONS OVER 50 INDEPENDENT RUNS OF MAEA-CmOPs AND COMPARISON WITH OTHER ALGORITHMS FOR $f_1 \sim f_4$

		MAEA-CmOPs	[26]	[29]	[28]
f_1	n=30	**799**	—	—	419 687
	n=60	**3578**	—	—	—
	n=90	**8802**	—	—	—
f_2	n=30	**842**	8500	6510	—
	n=60	**3817**	27 300	25 200	—
	n=90	**9790**	57 000	45 300	—
f_3	n=30	**869**	14 300	7150	—
	n=60	**4088**	41 250	39 620	—
	n=90	**8956**	75 450	67 620	—
f_4	n=30	**2098**	9000	3150	—
	n=60	16 099	36 000	**13 010**	—
	n=90	50 260	45 900	**24 310**	—

cost needs. Therefore, it is set to 2 to save the computational cost. Finally, the stop criterion is set to find out a global optimum solution.

A. Strong-Linkage Deceptive Functions

The four strong-linkage deceptive functions used here are

$$f_1(\boldsymbol{a}) = \sum_{i=1}^{n/3} f_{\text{Goldberg3}}(a_{3i-2}, a_{3i-1}, a_{3i})$$
$$f_2(\boldsymbol{a}) = \sum_{i=1}^{n/3} f_{\text{deceptive3}}(a_{3i-2}, a_{3i-1}, a_{3i})$$
$$f_3(\boldsymbol{a}) = \sum_{i=1}^{n/5} f_{\text{trap5}}(a_{5i-4}, a_{5i-3}, a_{5i-2}, a_{5i-1}, a_{5i})$$
$$f_4(\boldsymbol{a}) = \sum_{i=1}^{n/6} f_{\text{bipolar6}}(a_{6i-5}, a_{6i-4}, a_{6i-3}, a_{6i-2}, a_{6i-1}, a_{6i}). \tag{25}$$

The experimental results in terms of the average number of function evaluations over 50 independent runs of MAEA-CmOPs when $n = 30, 60,$ and 90 are given in Table I and are also compared with those in [26], [28], and [29]. The results show that, for f_1, f_2, and f_3, the computational cost of MAEA-CmOPs is far smaller than those of the other algorithms and is only about 10%–20% of those in [26] and [29]. For f_4, when $n = 30$, MAEA-CmOP outperforms the other methods, and when $n = 60$ and 90, MAEA-CmOP outperforms the method in [26] but is outperformed by the method in [29].

To further validate MAEA-CmOP's performance, particularly the performance in processing large-scale problems, the following experiments are done: for $f_1 \sim f_3$, n increases from 30 to 990 in steps of 60; and for f_4, n increases from 30 to 210 in steps of 30. For each value of n, 50 independent runs of MAEA-CmOP are done, and the average number of function evaluations is shown in Fig. 2.

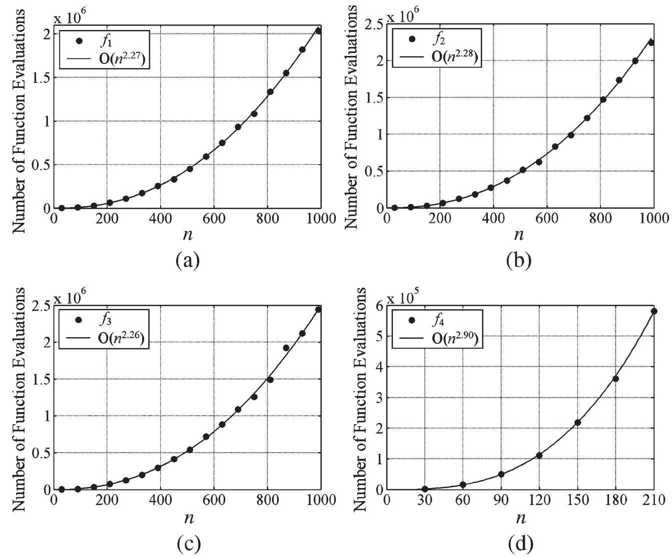

Fig. 2. Number of function evaluations increasing with the problem size of MAEA-CmOPs for strong-linkage deceptive functions.

TABLE II
COMPARISON IN TERMS OF THE AVERAGE NUMBER OF FUNCTION EVALUATIONS FOR f_2 AND f_3 BETWEEN MAEA-CmOPs AND THE METHOD IN [30]

		MAEA-CmOPs	[30]
f_2	n=60	**3817**	28 807
	n=240	**96 061**	235 126
	n=510	**516 144**	—
	n=810	**1 470 082**	—
	n=990	**2 244 465**	—
f_3	n=100	**12 514**	97 746
	n=250	**99 899**	478 410
	n=510	**541 398**	—
	n=810	**1 489 900**	—
	n=990	**2 443 265**	—

As can be seen, the time complexities of $f_1 \sim f_4$ can be approximated by $(0.33 \times n^{2.27})$, $(0.34 \times n^{2.28})$, $(0.41 \times n^{2.26})$, and $(0.11 \times n^{2.90})$, respectively. The coefficients of all these four approximate functions are smaller than 1. The exponentials of $f_1 \sim f_3$ are less than 2.28, and that of f_4 is a bit larger, namely, 2.90. In general, for all these four functions, the time complexity of MAEA-CmOP increases in a polynomial basis with the problem size.

Similar experiments have been done in [30] for f_2 and f_3. In [30], n increases from 60 to 240 for f_2 and from 100 to 250 for f_3. Thus, a comparison between MAEA-CmOP and the method in [30] is given in Table II, where the results of the method in [30] are obtained by their software.[1] Table II shows that the computational cost of MAEA-CmOP is far smaller than that of the method in [30] and is only about 10%–40% of the computational cost of the method in [30]. In addition, even when n increases to 990, the computational cost of MAEA-CmOP is still less than 2.5 million function evaluations.

[1] http://www-illigal.ge.uiuc.edu/sourcecd.html.

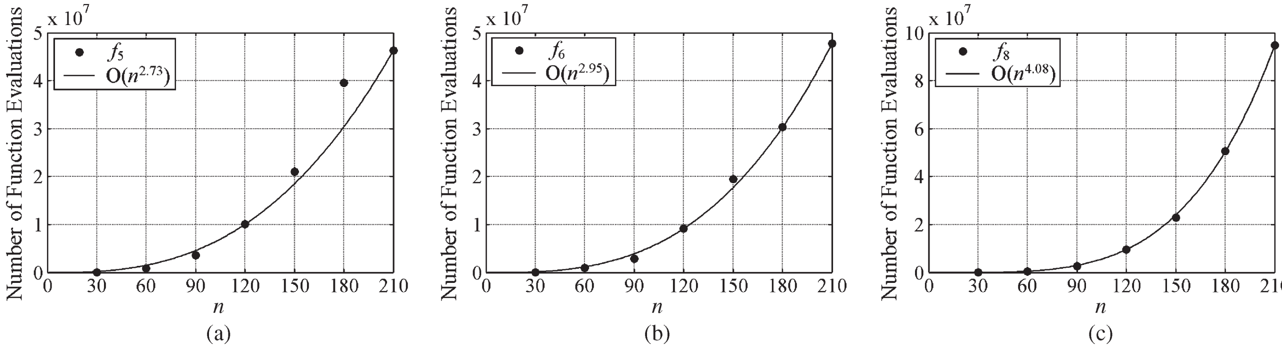

Fig. 3. Number of function evaluations increasing with the problem size of MAEA-CmOPs for weak-linkage deceptive functions.

TABLE III
COMPARISON BETWEEN MAEA-CmOPs FOR WEAK-LINKAGE AND STRONG-LINKAGE DECEPTIVE FUNCTIONS

		f_1	f_5	f_2	f_6	f_3	f_7	f_4	f_8
Average number of function evaluations	n=30	799	67 290	842	69 541	869	4 552 01	2098	60 111
	n=60	3578	905 207	3817	929 707	4088	428 605 105	16 099	1 578 582
	n=90	8802	3 589 483	9790	2 851 883	8956	—	50 260	8 027 175
	n=210	61 613	46 318 640	67 205	47 719 535	73 845	—	1 742 155	284 445 620
Time complexity	Coefficients	0.33	21.60	0.34	6.82	0.41	—	0.11	0.03
	Exponentials	O($n^{2.27}$)	O($n^{2.73}$)	O($n^{2.28}$)	O($n^{2.95}$)	O($n^{2.26}$)	—	O($n^{2.90}$)	O($n^{4.08}$)

B. Weak-Linkage Deceptive Functions

The four weak-linkage deceptive functions used here are

$$f_5(\boldsymbol{a}) = \sum_{i=1}^{n/3} f_{\text{Goldberg3}}(a_i, a_{i+n/3}, a_{i+2n/3})$$

$$f_6(\boldsymbol{a}) = \sum_{i=1}^{n/3} f_{\text{deceptive3}}(a_i, a_{i+n/3}, a_{i+2n/3})$$

$$f_7(\boldsymbol{a}) = \sum_{i=1}^{n/5} f_{\text{trap5}}(a_i, a_{i+n/5}, a_{i+2n/5}, a_{i+3n/5}, a_{i+4n/5})$$

$$f_8(\boldsymbol{a}) = \sum_{i=1}^{n/6} f_{\text{bipolar6}}(a_i, a_{i+n/6}, a_{i+2n/6}, a_{i+3n/6}, a_{i+4n/6}, a_{i+5n/6}). \tag{26}$$

The experiments in this section are designed as follows: for f_5, f_6, and f_8, n increases from 30 to 210 in steps of 30, and 50 independent runs of MAEA-CmOP are done on each selected n. For f_7, since the computational cost is too high, only the experiments when $n = 30$ and 60 are done. The average number of function evaluations for f_5, f_6, and f_8 is given in Fig. 3.

As can be seen, the time complexities of f_5, f_6, and f_8 can be approximated by $(21.60 \times n^{2.73})$, $(6.82 \times n^{2.95})$, and $(0.03 \times n^{4.08})$, respectively. Therefore, for these three functions, the time complexity of MAEA-CmOP still increases in a polynomial basis with the problem size. For f_8, although the exponential of the approximate function is larger, the coefficient is small, i.e., only 0.03. Apart from this, the number of function evaluations of the method in [28] for f_5 with $n = 30$ is 421 401, whereas that of MAEA-CmOPs is 67 290 and is far smaller than that of the method in [28].

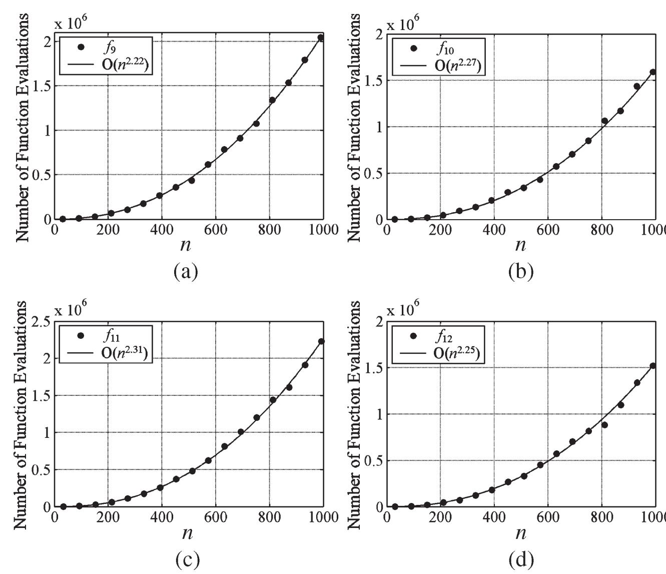

Fig. 4. Number of function evaluations increasing with the problem size of MAEA-CmOPs for overlapping-linkage deceptive functions.

In fact, the subfunctions of these four weak-linkage functions are the same as those of the above four strong-linkage functions, which are changed in the way the variables relate to each other. According to the schema theorem and the building block assumption, a weak-linkage function is more difficult than the corresponding strong-linkage function. Thus, a comparison is made between the results of MAEA-CmOP for strong-linkage and weak-linkage functions in Table III.

As can be seen, from the viewpoint of the number of function evaluations, weak-linkage deceptive functions need far more function evaluations than the corresponding strong-linkage deceptive functions. From the viewpoint of the time complexity, f_1 and f_5, f_2 and f_6 are similar, but the coefficients of weak-linkage functions are larger than those of strong-linkage

TABLE IV
COMPARISON BETWEEN MAEA-CmOPs FOR STRONG-LINKAGE AND OVERLAPPING-LINKAGE DECEPTIVE FUNCTIONS

		f_2	f_9	f_{10}	f_3	f_{11}	f_{12}
Average number of function evaluations	n=30	842	852	831	869	1007	809
	n=60	3817	3880	2931	4088	3833	3144
	n=90	9790	9746	7211	8956	10 477	7159
	n=510	516 144	435 705	341 468	541 398	479 595	329 751
	n=810	1 470 082	1 339 605	1 064 319	1 489 900	1 439 348	883 164
	n=990	2 244 465	2 044 992	1 589 103	2 443 265	2 227 592	1 521 552
Time complexity	Coefficients	0.34	0.46	0.26	0.41	0.26	0.27
	Exponentials	O($n^{2.28}$)	O($n^{2.22}$)	O($n^{2.27}$)	O($n^{2.26}$)	O($n^{2.31}$)	O($n^{2.25}$)

functions. In general, weak-linkage deception functions are more difficult than strong-linkage ones.

C. Overlapping-Linkage Deceptive Functions

The four overlapping-linkage deceptive functions used here are

$$
\begin{aligned}
f_9(\boldsymbol{a}) &= \sum_{i=1}^{\frac{n-1}{2}} f_{\text{deceptive3}}(a_{2i-1}, a_{2i}, a_{2i+1}) \\
f_{10}(\boldsymbol{a}) &= \sum_{i=1}^{n-2} f_{\text{deceptive3}}(a_i, a_{i+1}, a_{i+2}) \\
f_{11}(\boldsymbol{a}) &= \sum_{i=1}^{\frac{n-1}{4}} f_{\text{trap5}}(a_{4i-3}, a_{4i-2}, a_{4i-1}, a_{4i}, a_{4i+1}) \\
f_{12}(\boldsymbol{a}) &= \sum_{i=1}^{\frac{n-3}{2}} f_{\text{trap5}}(a_{2i-1}, a_{2i}, a_{2i+1}, a_{2i+2}, a_{2i+3})
\end{aligned} \quad (27)
$$

where there is one overlapping variable between two successive subfunctions in f_9 and f_{11}, two overlapping variables in f_{10}, and three overlapping variables in f_{12}. The experiments in this section are designed as follows: For $f_9 \sim f_{12}$, n increases from 30 to 990 in steps of 60, and 50 independent runs of MAEA-CmOP are done on each selected n, and the results are shown in Fig. 4.

As can be seen, the time complexities of $f_9 \sim f_{12}$ can be approximated by $(0.46 \times n^{2.22})$, $(0.26 \times n^{2.27})$, $(0.26 \times n^{2.31})$, and $(0.27 \times n^{2.25})$, respectively. The approximation functions of these four overlapping-linkage functions are similar to those of f_2 and f_3, that is, all the coefficients are smaller than 1, all exponentials are less than 2.31, and the time complexity increases in a polynomial basis with the problem size. Apart from this, the number of function evaluations of the method in [29] for f_9 with $n = 30$, 60, and 90 are 14 710, 40 270, and 76 120, respectively. While those of MAEA-CmOP are 852, 3880, and 9746, respectively, and are far smaller than those of the method in [29].

In fact, f_2, f_9, and f_{10} use the same subfunctions, and f_3, f_{11}, and f_{12} use the same subfunctions, and only the way the variables relate to each other is different. Therefore, a comparison between the performances of MAEA-CmOP on strong-linkage and overlapping-linkage deceptive functions is given in Table IV.

As can be seen, in general, the difficulty of overlapping-linkage deceptive functions is similar to that of the corresponding strong-linkage deceptive functions in both the number of function evaluations and the time complexity. Moreover, the temporal cost incurred by MAEA-CmOP keeps in the range of 1.5–2.5 million function evaluations, even for functions with 990 dimensions.

V. EXPERIMENTS ON HIERARCHICAL PROBLEMS

Many problems in business, engineering, and science have a hierarchical structure. By hierarchy, we mean a system consisting of subsystems, each of which is a hierarchy by itself, until we reach some bottom level. The interactions within each subsystem are of much higher magnitude than the interactions between the subsystems. There are plenty of hierarchy examples around us. A university consists of colleges, colleges consist of departments, departments consist of laboratories and offices, and so forth. A program code consists of procedures and functions, procedures consist of single commands and library calls, commands consist of machine code or assembly language, and so forth [30].

These hierarchical problems do not like the above deceptive functions, which can be decomposed into independent subfunctions; instead, their functions interact with each other and form a treelike hierarchical structure. To study the performance of EAs in solving this kind of problem, Pelikan [30] made a thorough research and designed two deceptive functions with hierarchical structure. Thus, these two deceptive functions and the famous hierarchical if-and-only-if (HIFF) function [31] are used to test the performance of MAEA-CmOPs in this section.

A. Hierarchical Problems

A hierarchical problem consists of a structure, a mapping function, and a function value. The input variables are at the lowest level, and the mapping function maps a lower level to an upper level, with a treelike structure resulting, and the sum of function values in each level consisting the final function value. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \boldsymbol{\mathcal{S}}$, and the variables in the ith level consisting of $\boldsymbol{a}^i = (a_1^i, a_2^i, \ldots, a_{n^i}^i)$, where $\boldsymbol{a}^1 = \boldsymbol{a}$, and the mapping function is f^{mapping}.

1) HIFF Function [31]: The mapping function maps two variables in the lower level and one variable in the upper level, and the problem size must satisfy $n = 2^{\text{Level}}$, and the number

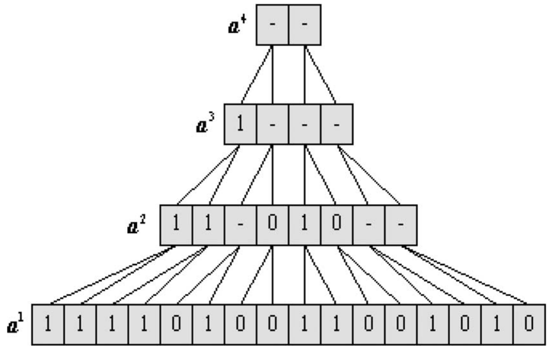

Fig. 5. Mapping process of the mapping function in HIFF function for the example with $n = 16$.

of variables in each level satisfies $(n^1 = n)$, $(n^i \times 2 = n^{i-1})$, $i = 2, 3, \ldots, \text{Level}$. The mapping function is given as

$$f_{\text{HIFF}}^{\text{mapping}}\left(a_j^i\right) = \begin{cases} 0, & \left(a_{2j-1}^{i-1} = 0\right) \text{ and } \left(a_{2j}^{i-1} = 0\right) \\ 1, & \left(a_{2j-1}^{i-1} = 1\right) \text{ and } \left(a_{2j}^{i-1} = 1\right) \\ -, & \text{otherwise} \end{cases} \quad (28)$$

where $i = 2, 3, \ldots, \text{Level}$, $j = 1, 2, \ldots, n^i$, and an example with $n = 16$ for the mapping process is also given in Fig. 5.

The function value in each level is computed by

$$f_{\text{HIFF}}^i(\boldsymbol{a}^i) = 2^{i-1} \sum_{j=1}^{n^i} f_j^i \quad (29)$$

where $f_j^i = \begin{cases} 1, & \left(a_j^i = 0\right) \text{ or } \left(a_j^i = 1\right) \\ 0, & \text{otherwise} \end{cases}$, $i = 1, 2, \ldots, Level$.

The final function value is computed by

$$f_{\text{HIFF}}(\boldsymbol{a}) = \sum_{i=1}^{\text{Level}} f_{\text{HIFF}}^i(\boldsymbol{a}^i) + \begin{cases} 2^{\text{Level}}, & \left(a_1^{\text{Level}} = a_2^{\text{Level}} = 0\right) \\ & \text{or } \left(a_1^{\text{Level}} = a_2^{\text{Level}} = 1\right) \\ 0, & \text{otherwise} \end{cases}. \quad (30)$$

2) Hierarchical Trap I [30]: The mapping function maps three variables in a lower level to one variable in an upper level, and the problem size must satisfy $n = 3^{\text{Level}}$, and the number of variables in each level satisfies $(n^1 = n)$, $(n^i \times 3 = n^{i-1})$, $i = 2, 3, \ldots, \text{Level}$. The mapping function is given as

$$f_{\text{HtrapI}}^{\text{mapping}}\left(a_j^i\right) = \begin{cases} 0, & \left(a_{3j-2}^{i-1} = 0\right) \text{ and } \left(a_{3j-1}^{i-1} = 0\right) \text{ and } \left(a_{3j}^{i-1} = 0\right) \\ 1, & \left(a_{3j-2}^{i-1} = 1\right) \text{ and } \left(a_{3j-1}^{i-1} = 1\right) \text{ and } \left(a_{3j}^{i-1} = 1\right) \\ -, & \text{otherwise} \end{cases} \quad (31)$$

where $i = 2, 3, \ldots, \text{Level}$, $j = 1, 2, \ldots, n^i$, and an example with $n = 27$ for the mapping process is also given in Fig. 6.

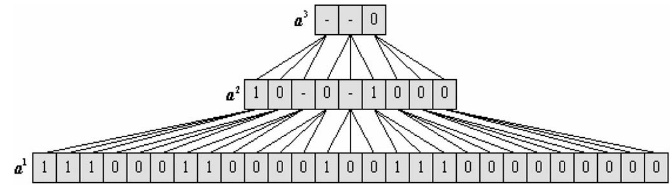

Fig. 6. Mapping process of the mapping function in Hierarchical trap I for the example with $n = 27$.

The function values of levels 1 to $(Level - 1)$ are computed as follows, where u denotes the number of 1's in $(a_{3j-2}^i, a_{3j-1}^i, a_{3j}^i)$:

$$f_{\text{HtrapI}}^i(\boldsymbol{a}^i) = 3^i \sum_{j=1}^{n^i/3} f_j^i\left(a_{3j-2}^i, a_{3j-1}^i, a_{3j}^i\right) \quad (32)$$

where

$$f_j^i\left(a_{3j-2}^i, a_{3j-1}^i, a_{3j}^i\right) = \begin{cases} 1, & (u = 3) \text{ or } (u = 0) \\ 0, & u = 2 \\ 0.5, & u = 1 \end{cases}.$$

The final function value is computed as follows, where u denotes the number of 1's in $(a_1^{\text{Level}}, a_2^{\text{Level}}, a_3^{\text{Level}})$:

$$f_{\text{HtrapI}}(\boldsymbol{a}) = \sum_{i=1}^{\text{Level}-1} f_{\text{HtrapI}}^i(\boldsymbol{a}^i) + 3^{\text{Level}} \times \begin{cases} 1, & u = 3 \\ 0, & u = 2 \\ 0.45, & u = 1 \\ 0.9, & u = 0 \end{cases}. \quad (33)$$

3) Hierarchical Trap II [30]: The structure and the mapping function are the same of that of Hierarchical Trap I. The function values of levels 1 to $(\text{Level} - 1)$ are computed as follows, where u denotes the number of 1's in $(a_{3j-2}^i, a_{3j-1}^i, a_{3j}^i)$:

$$f_{\text{HtrapII}}^i(\boldsymbol{a}^i) = 3^i \sum_{j=1}^{n^i/3} f_j^i\left(a_{3j-2}^i, a_{3j-1}^i, a_{3j}^i\right) \quad (34)$$

where

$$f_j^i\left(a_{3j-2}^i, a_{3j-1}^i, a_{3j}^i\right) = \begin{cases} 1, & u=3 \\ 1+0.05/\text{Level}-u/2, & \text{otherwise} \end{cases}.$$

The final function value is computed as follows, where u denotes the number of 1's in $(a_1^{\text{Level}}, a_2^{\text{Level}}, a_3^{\text{Level}})$:

$$f_{\text{HtrapII}}(\boldsymbol{a}) = \sum_{i=1}^{\text{Level}-1} f_{\text{HtrapII}}^i(\boldsymbol{a}^i) + 3^{\text{Level}} \times \begin{cases} 1, & u = 3 \\ 0, & u = 2 \\ 0.45, & u = 1 \\ 0.9, & u = 0 \end{cases}. \quad (35)$$

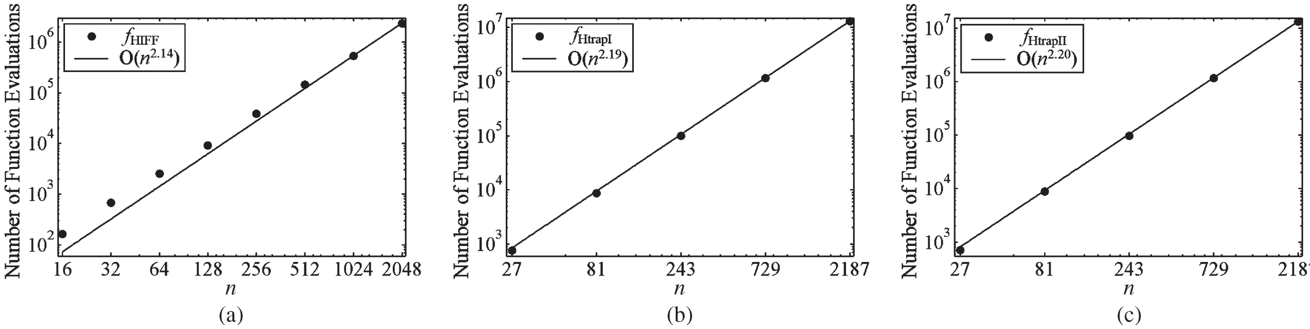

Fig. 7. Number of function evaluations increasing with the problem size of MAEA-CmOPs for hierarchical problems.

TABLE V
COMPARISON IN TERMS OF THE NUMBER OF FUNCTION EVALUATIONS IN SOLVING f_{HIFF}, f_{HtrapI}, AND f_{HtrapII} BETWEEN MAEA-CmOPs AND THE METHOD IN [30]

		MAEA-CmOPs	[30]
f_{HIFF}	n=128	**9219**	26 491
	n=256	**38 340**	88 859
	n=512	**143 859**	400 498
	n=1024	**531 151**	—
	n=2048	**2 339 951**	—
f_{HtrapI}	n=27	**758**	3433
	n=81	**8641**	29 127
	n=243	**98 471**	221 411
	n=729	**1 166 761**	1 428 236
	n=2187	**12 980 812**	—
f_{HtrapII}	n=27	**708**	5058
	n=81	**8760**	35 645
	n=243	**97 422**	237 684
	n=729	**1 169 227**	—
	n=2187	**13 172 477**	—

Apparently, the global optimum solutions of f_{HIFF} are the vectors with all values equal to 1 or 0, whereas those of f_{HtrapI} and f_{HtrapII} are the vectors with all values equal to 1.

B. Experiments and Analyses

The parameters of MAEA-CmOPs in this section are the same of those in Section IV, and the experiments are designed as follows: For f_{HIFF}, n increases from 16 (2^4) to 2048 (2^{11}), and for f_{HtrapI} and f_{HtrapII}, n increases from 27 (3^3) to 2187 (3^7). Fifty independent runs of MAEA-CmOPs are done on each selected n. The number of function evaluations is shown in Fig. 7.

As can be seen, the time complexities of f_{HIFF}, f_{HtrapI}, and f_{HtrapII} can be approximated by $(0.19 \times n^{2.14})$, $(0.62 \times n^{2.19})$, and $(0.57 \times n^{2.20})$, respectively. The time complexities of MAEA-CmOPs in solving these three functions are similar, that is, all the coefficients of the approximation functions are smaller than 1, whereas all the exponentials are less than 2.20.

Similar experiments have been done in [30] for f_{HIFF}, f_{HtrapI}, and f_{HtrapII}, where n increases from 16 to 512 for f_{HIFF}, and from 27 to 729 for f_{HtrapI}, and from 27 to 243 for f_{HtrapII}. Thus, a comparison between MAEA-CmOPs and the method in [30] is given in Table V.

As can be seen, the computational cost of MAEA-CmOPs is far smaller than that of the method in [30] and is only 20%–40% of that of the method in [30]. In addition, for f_{HIFF}, when the problem size increases to 2048, the temporal cost incurred by MAEA-CmOPs only reaches 2.3 million function evaluations; and for f_{HtrapI} and f_{HtrapII}, when the problem size increases to 2187, although the computational cost is a bit larger, it is still about 13 million function evaluations. All these results show that MAEA-CmOP obtains a good performance in solving large-scale hierarchical problems.

VI. CONCLUSION

Multiagent systems and EAs have been integrated in this paper, and the agent behaviors are realized by means of evolution. Thus, a new algorithm for CmOPs, namely, MAEA-CmOPs, has been proposed, and its convergence is analyzed theoretically. In the experiments, strong-linkage, weak-linkage, and overlapping-linkage deceptive functions and hierarchical problems with treelike structure are used to test the performance of MAEA-CmOPs comprehensively, and large-scale problems whose dimensions are more than 1000 are used to study the time complexity of MAEA-CmOPs.

The experimental results show that deceptive functions have different difficulties, depending on the way to connect the subfunctions. For MAEA-CmOPs, strong-linkage and overlapping-linkage deceptive functions consisting of three- and five-order deceptive functions have the same time complexity, namely, $O(n^{2.2}) \sim O(n^{2.3})$, whereas bipole deceptive and weak-linkage functions are more difficult since the time complexity is $O(n^{2.7}) \sim O(n^{4.0})$. For hierarchical problems, the time complexity of MAEA-CmOPs is $O(n^{2.1}) \sim O(n^{2.2})$.

To summarize, MAEA-CmOP obtains a polynomial time complexity for all test problems. In addition, the parameters of MAEA-CmOPs are simple and easy to be tuned. All of the experimental results are obtained under the same parameter settings, which illustrates that MAEA-CmOP is robust and easy to use.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful comments and valuable suggestions.

REFERENCES

[1] J. H. Holland, *Adaptation in Nature and Artificial System*. Cambridge, MA: MIT Press, 1992.

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996. 3rd Revised and Extended.

[3] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1998. Reprint ed.

[4] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, U.K.: Oxford Univ. Press, 1996.

[5] D. B. Fogel, *Evolutionary Computation: The Fossil Record*, 1st ed. New York: Wiley-IEEE Press, 1998.

[6] E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 1st ed. ser. Natural Computing Series. New York: Springer-Verlag, 2003.

[7] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A multiagent genetic algorithm for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1128–1141, Apr. 2004.

[8] J. Liu, W. Zhong, and L. Jiao, "A multiagent evolutionary algorithm for constraint satisfaction problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 54–73, Feb. 2006.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[10] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3rd ed. New York: Wiley-IEEE Press, 2005.

[11] J. Liu, W. Zhong, L. Jiao, and X. Li, "Moving block sequence and organizational evolutionary algorithm for general floorplanning with arbitrarily shaped rectilinear blocks," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 630–646, Oct. 2008.

[12] J. Liu, W. Zhong, and L. Jiao, "An organizational evolutionary algorithm for numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 4, pp. 1052–1064, Aug. 2007.

[13] L. Jiao, J. Liu, and W. Zhong, "An organizational coevolutionary algorithm for classification," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 67–80, Mar. 2006.

[14] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. New York: Addison-Wesley, 1999.

[15] J. Liu, *Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation*. Singapore: World Scientific, 2001.

[16] J. Liu, Y. Y. Tang, and Y. C. Cao, "An evolutionary autonomous agents approach to image feature extraction," *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 141–158, Jul. 1997.

[17] J. Liu, H. Jing, and Y. Y. Tang, "Multi-agent oriented constraint satisfaction," *Artif. Intell.*, vol. 136, no. 1, pp. 101–144, Mar. 2002.

[18] P. T. Sandanayake and D. J. Cook, "ONASI: Online agent modeling using a scalable Markov model," *Int. J. Pattern Recogn. Artif. Intell.*, vol. 17, no. 5, pp. 757–779, 2003.

[19] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*. New York: Springer-Verlag, 2008.

[20] D. Whitley, "Cellular genetic algorithms," in *Proc. 5th Int. Conf.* Genetic *Algorithms*, R. K. Belew and L. B. Booker, Eds, 1993, p. 658.

[21] T. Nakashima, T. Ariyama, T. Yoshida, and H. Ishibuchi, "Performance evaluation of combined cellular genetic algorithms for function optimization problems," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, Kobe, Japan, 2003, vol. 1, pp. 295–299.

[22] G. Folino, C. Pizzuti, and G. Spezzano, "Parallel hybrid method for SAT that couples genetic algorithms and local search," *IEEE Trans. Evol. Comput.*, vol. 5, no. 4, pp. 323–334, Aug. 2001.

[23] M. Iosifescu, *Finite Markov Processes and Their Applications*. Chichester, U.K.: Wiley, 1980.

[24] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96–101, Jan. 1994.

[25] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms revisited: Studies in mixed size and scale," *Complex Syst.*, vol. 4, no. 4, pp. 415–444, 1990.

[26] M. Pelikan and D. E. Goldberg, "BOA: The Bayesian optimization algorithm," Illinois Genetic Algorithms Lab.. Univ. Illinois, Urbana-Champaign, Urbana, IL, IlliGAL Rep. 98013, 1998.

[27] K. Deb and D. E. Goldberg, "Analyzing deception in trap functions," Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana-Champaign, Urbana, IL, IlliGAL Rep. 91009, 1991.

[28] S. Wu, Q. Zhang, and H. Chen, "A new evolutionary algorithm based on family eugenics," *J. Softw.*, vol. 8, no. 2, pp. 137–144, 1997. (in Chinese).

[29] Y. Lin and X. Yang, "Research on fast evolutionary algorithms based on probabilistic models," *Acta Electronica Sinica*, vol. 29, no. 2, pp. 178–181, 2001. (in Chinese).

[30] M. Pelikan, *Bayesian Optimization Algorithm: From Single Level to Hierarchy*. Urbana, IL: Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana-Champaign, 2002.

[31] R. A. Watson, G. S. Hornby, and J. B. Pollack, *Modeling Building-Block Interdependency*, vol. 1498. Berlin, Germany: Springer-Verlag, 1998, pp. 97–106.

Jing Liu (M'06) was born in Xi'an, China. She received the B.S. degree in computer science and technology and the Ph.D. degree in circuits and systems from Xidian University, Xi'an, in 2000 and 2004, respectively.

She is currently a Professor with Xidian University. Her research interests include evolutionary computation, multiagent systems, and data mining.

Weicai Zhong (M'06) was born in Jiangxi, China. He received the B.S. degree in computer science and technology and the Ph.D. degree in pattern recognition and intelligent information system from Xidian University, Xi'an, China, in 2000 and 2004, respectively.

He is currently a Postdoctoral Fellow with Xidian University. His research interests include evolutionary computation, data mining, and statistical learning.

Licheng Jiao (SM'89) was born in Shaanxi, China, on October 15, 1959. He received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1982 and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively.

From 1984 to 1986, he was an Assistant Professor with the Civil Aviation Institute of China, Tianjing, China. During 1990 and 1991, he was a Postdoctoral Fellow with the National Key Lab for Radar Signal Processing, Xidian University, Xi'an. He is currently the Dean of the School of Electronic Engineering and the Director of the Institute of Intelligent Information Processing, Xidian University. He is the author of three books: *Theory of Neural Network Systems* (Xi'an, China: Xidian University Press, 1990), *Theory and Application on Nonlinear Transformation Functions* (Xi'an, China: Xidian University Press, 1992), and *Applications and Implementations of Neural Networks* (Xi'an, China: Xidian University Press, 1996). He is the author or coauthor of more than 150 scientific papers. His current research interests include signal and image processing, nonlinear circuit and system theory, learning theory and algorithms, optimization problems, wavelet theory, and data mining.