Front. Electr. Electron. Eng. 2012, 7(1): 16–31
DOI 10.1007/s11460-012-0192-0

REVIEW ARTICLE

Jun ZHANG, Wei-Neng CHEN, Zhi-Hui ZHAN, Wei-Jie YU, Yuan-Long LI, Ni CHEN, Qi ZHOU

A survey on algorithm adaptation in evolutionary computation

Abstract Evolutionary computation (EC) is one of the fastest growing areas in computer science that solves intractable optimization problems by emulating biologic evolution and organizational behaviors in nature. To design an EC algorithm, one needs to determine a set of algorithmic configurations like operator selections and parameter settings. How to design an effective and efficient adaptation scheme for adjusting the configurations of EC algorithms has become a significant and promising research topic in the EC research community. This paper intends to provide a comprehensive survey on this rapidly growing field. We present a classification of adaptive EC (AEC) algorithms from the perspective of how an adaptation scheme is designed, involving the adaptation objects, adaptation evidences, and adaptation methods. In particular, by analyzing the population distribution characteristics of EC algorithms, we discuss why and how the evolutionary state information of EC can be estimated and utilized for designing effective EC adaptation schemes. Two AEC algorithms using the idea of evolutionary state estimation, including the clustering-based adaptive genetic algorithm and the adaptive particle swarm optimization algorithm are presented in detail. Some potential directions for the research of AECs are also discussed in this paper.

Keywords evolutionary algorithm (EA), evolutionary computation (EC), algorithm adaptation, parameter control

Received October 14, 2011; accepted December 27, 2011

Jun ZHANG (✉), Wei-Neng CHEN, Zhi-Hui ZHAN, Wei-Jie YU, Yuan-Long LI, Ni CHEN, Qi ZHOU
Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China
Key Laboratory of Digital Life, Ministry of Education, Guangzhou 510275, China
Key Laboratory of Software Technology, Education Department of Guangdong Province, Guangzhou 510275, China
E-mail: junzhang@ieee.org

1 Introduction

Evolutionary computation (EC) algorithms are a class of optimization techniques inspired by biologic evolution and organizational behaviors of living creatures. Generally, EC algorithms include genetic algorithm (GA), evolutionary programming (EP), evolutionary strategies (ES), differential evolution (DE), estimation of distribution algorithm (EDA), particle swarm optimization (PSO), ant colony optimization (ACO), and memetic algorithm (MA). Instead of providing all details for implementation, the description of a specific EC algorithm usually leaves a number of factors undetermined. These factors, known as the configurations of ECs, include the setting of a particular parameter, the implementation of an operator, the structure of the population, etc. Investigations have shown that these algorithm configurations are crucial to the performance and behavior of ECs [1–4].

Finding appropriate configurations for ECs has long been an interesting and significant research topic in the EC community. Traditionally, fixed settings are applied to ECs. The algorithm configurations (e.g., the parameters and operators) are determined before execution according to prior guidelines or are tuned manually [2,5]. However, there are drawbacks with fixed settings. Both theoretical and empirical studies have shown that the optimal configurations of an EC algorithm are specific to the optimization problems [6]. Predefined guidelines are inadequate for that they cannot be applied to numerous different application problems. Besides, the most beneficial algorithm configurations of ECs might be different in different stages of evolution [1,5,7–9]. The above facts indicate that fixed algorithm configurations cannot satisfy the requirements of optimization.

To overcome the drawbacks of fixed settings, much research attention has been paid on controlling the EC configurations dynamically, among which the algorithm parameters control is the most popular research topic in most of the existing work. In Ref. [10], the parameter

control strategies in ECs are classified into three categories, namely, the deterministic control, the adaptive control, and the self-adaptive control. The deterministic control modifies the parameters according to deterministic rules, and utilizes no feedback from the evolution. In contrast, both the adaptive control and the self-adaptive control are related to certain form of feedback of the algorithm. The latter two strategies are preferred to deterministic control for that they consider the actual progress of evolution. In this paper, both adaptive control and self-adaptive control are regarded as "adaptive" since the latter can be regarded as a subset of the former.

Various adaptive ECs (AECs) with controlled configurations have been developed, and the results of empirical studies have shown to be encouraging. For all AECs, three issues should be addressed when designing adaptation schemes. First, we have to determine what objects in the AEC are to be adapted. That is, whether the parameters, operators, or population structures are to be adaptively controlled? Second, we have to determine what evidences are used to activate the adaptation. That is, when to execute the adaptation? The evidence includes different forms of information on the evolutionary process, e.g., fitness values, positional distribution of population, or both of them. Third, we have to determine what methods are used for controlling the EC algorithm configurations.

In this paper, we conduct a comprehensive survey on the design of adaptation scheme in ECs. Corresponding to the above-stated three issues, the adaptation schemes are classified according to three different taxonomies, i.e., the objects of control, the evidences of control, and the methods of control. To provide a more insightful view on the topic, we focus on a class of advanced AEC based on estimation of the evolutionary state. To help better understandings on their effectiveness, we will explain the motivation of introducing the "evolutionary state", followed by detailed descriptions of the adaptation schemes in an adaptive GA (AGA) [11] and an adaptive PSO (APSO) [12]. Besides, potential research directions in the AEC research area are discussed.

The rest of this paper is organized as follows. Section 2 presents classification of AECs according to three different taxonomies. Two advanced AECs based on estimation of evolutionary state are introduced in Section 3. In Section 4, potential research trends for the AECs are discussed. Conclusions are finally drawn in Section 5.

2 Classification of adaptation schemes

When designing an adaptation scheme for ECs, three aspects need to be taken into considerations. 1) We have to decide the component to be adapted in an EC algorithm, i.e., the adaptation objects. Major components of an EC algorithm contain control parameters, evolutionary operators, population structure, etc. 2) We should decide what is the adaptation based on, i.e., the adaptation evidences. There are many kinds of evidences that can be used for adaptation. Among them, deterministic factors, fitness values of individuals, and population distribution are most commonly used. 3) After determining the evidence for adaptation, finally we need to design an adaptation method to perform the adaptive control. The adaptation methods may be designed based on some simple rules, or co-evolving the adaptation components with the population. Other methods such as entropy-based control and fuzzy control methods are also frequently used. In the following contexts, we classify the adaptation schemes from the above perspectives and discuss each item in detail. The classification can be illustrated by Fig. 1.

2.1 Adaptation objects

The first criterion for classification naturally concerns the adaptation objects. Considering the major components of an EC, we classify the adaptation objects into four categories as control parameters, evolutionary operator, population structure, and others.

1) Control parameters

An EC algorithm may be associated with a set of control parameters. Examples of control parameters in different EC algorithms are the crossover probability p_{x} and mutation probability p_{m} in GA [13,14], the inertia weight w and acceleration coefficients c_1, c_2 in PSO [15], and the pheromone evaporation rate ρ in ACO [16,17]. Table 1 summarizes the major control parameters in different ECs. These control parameters have significant effects on the optimization performance of the algorithms. This is because the values of them greatly influence the convergence speed and population diversity. However, it is a difficult task to choose the right parameter values. First, different types of problems may need different suitable parameter settings, and these control parameters are not independent but they interact with each other. Thus, it is time-consuming and impractical to find an optimal parameter setting empirically for a specific problem. Second, different stages of evolution may require different parameter values to achieve the best performance. For example, in the PSO algorithm, using a large inertia weight w in the early stage is helpful to explore new regions of the search space, and using a small inertia weight w in the late stage is good at exploiting the promising solutions. Therefore, it is more appropriate to adaptively control the parameter values during the

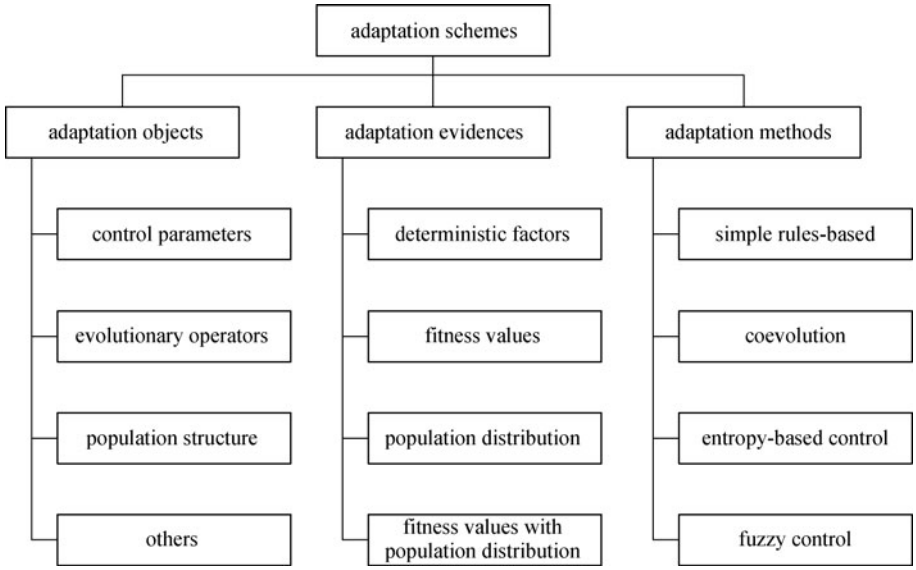


Fig. 1 Classification of adaptation schemes

Table 1 Major control parameters in different EC algorithms

algorithms	major control parameters
genetic algorithm (GA)	crossover probability p_{x}, mutation probability p_{m}
evolutionary programming (EP)	mutation probability p_{m}
evolutionary strategies (ES)	mutation step size σ
differential evolution (DE)	scale factor F, crossover probability CR
estimation of distribution algorithm (EDA)	probability model parameters
particle swarm optimization (PSO)	inertia weight w, acceleration coefficients c_1, c_2
ant colony optimization (ACO)	pheromone evaporation rate ρ
memetic algorithm (MA)	local learning strategy

evolutionary process.

2) Evolutionary operators

The procedure of an EC involves a sequence of evolutionary operators, e.g., the mutation, crossover, and selection in GA, EP, and DE, velocity and position updates in PSO, and solution construction in ACO. An evolutionary operator can usually be implemented in different ways. For example, there exist many mutation strategies in DE algorithms (e.g., DE/rand/1, DE/best/1, and DE/current-to-best/1). It is observed that different mutation strategies are required for different evolutionary states and different optimization problems to achieve the best DE performance [18]. For instance, the DE/rand/1 strategy is more suitable for exploration state and multimodal problems, while the DE/best/1 strategy is more suitable for exploitation state and unimodal problems. Therefore, the adaptation of mutation strategies in DE is desirable to make the algorithms less sensitive to different optimization scenarios. Another example is an elitist learning strategy (ELS) designed in the APSO proposed by Zhan et al. [12]. The ELS performs adaptively only in the convergence state of the optimization process. This approach is helpful to maintain population diversity and avoid premature convergence.

3) Population structure

The dynamic changes of population size and population topology are two main forms of the population adaptation in ECs. Determining an appropriate population size is usually a difficult task for ECs. Computational resources may be wasted if the population size is too large. In contrast, if the population size is too small, ECs may lose diversity too early and be trapped in local optima. Therefore, various methods have been proposed for adaptively controlling the population size for different ECs, such as GA [19], MA [20], DE [21,22], and multiobjective optimization algorithms [23]. Regarding the population topology, we take the PSO as an example. The main static topologies of PSO include global-best, local-best, pyramid, star, von Neumann, etc. [24]. Their effects on PSO have been systematically studied in Ref. [24]. Recent research suggests that adaptive topologies can be more competitive. The adaptive hierarchical PSO [25] and Frankenstein's PSO [26] which use time-varying population topologies are capable of performing better

than those using static ones.

4) Others

Apart from the above three categories of adaptive components, other adaptation objects have been made on the local search methods in MAs and migration strategy in parallel ECs, etc.

MA is one kind of hybrid ECs that combines global search and local search procedures. Extensive research has shown that the local search methods or memes in MAs greatly influence the search performance of the algorithms. Nevertheless, these memes are often problem-dependent and it consumes a lot of time to select one for a particular problem. To enhance the robustness of MAs, many adaptive MAs have been proposed, using multiple memes in the search and adaptively applying one on an individual. A detailed survey of memes adaptation in MAs can be found in Ref. [27].

Parallel ECs consist of multiple populations which communicate with each other usually by a migration process. The behaviors of these algorithms are affected by parameters such as migration rate, migration size, and communication topology. In Ref. [28], a pseudocoevolutionary GA using an adaptive migration strategy is designed for power electronic circuit (PEC) optimization. With the adaptive control of the migration frequency, the effectiveness and efficiency in optimizing the circuit parameters have been improved. Such an adaptive migration strategy was also applied to PSO by Zhan and Zhang [29], and the experimental results demonstrate its promising performance.

2.2 Adaptation evidences

After discussing which component of an EC algorithm can be adapted, we next consider what the adaptation can be based on. The evidence for adaptation can be classified into one of the four categories, i.e., deterministic factors, fitness values, population distribution, and combination of fitness value and population distribution.

1) Deterministic factors

The simplest evidence for algorithm adaptation may be some deterministic factors. These factors are not related to any information derived from the evolution. The generation number and fitness evaluations number are two such factors most commonly used. Due to their simplicity and low computational cost, a lot of control methods are developed based on these factors. For example, the inertia weight w of PSO [15] and scale factor F of DE [30] are linearly decreased based on the generation number. Merkle et al. [31] proposed an ACO algorithm for resource-constrained project scheduling problem. They decrease the value of β linearly after 50% of all generations, while start ρ with a small value and then set it to a larger value for the last 200 generations. As these simple deterministic factors can hardly capture the evolutionary behavior of EC algorithms, the effects of deterministic-factor-based adaptation schemes are usually limited.

2) Fitness values

Instead of using deterministic factors, other adaptations are made based on some form of feedback from the search. Since the improvement of fitness values are desired for all ECs, the fitness values of individuals are considered as one natural and direct form of feedback. The fitness values can be measured at two levels, i.e., the individual level and population level.

For the individual level, each individual often maintains its own parameters which are controlled according to the individual fitness value. In the adaptive DE (JADE) proposed by Zhang and Sanderson [32], each individual is associated with its own parameter F and CR. It is believed that better parameter values lead to better individuals and they should be propagated to the next generation. The parameter values for each individual are recorded if the fitness value of the individual is better than that of the previous generation. The parameter control process is then performed based on the record of parameter values that lead to better fitness values. Cai et al. [16] and Hao et al. [17] proposed adaptive ACOs for the traveling salesman problem. In their algorithms, each ant maintains its own value of parameter ρ, which is updated according to the quality of its solution. The motivation of the adaptation is that better solutions contribute more pheromone than the worse ones.

In other adaptation schemes, fitness values are measured at the population level and the parameters are also adapted for the whole population. For example, the adjustments of p_{x} and p_{m} in Refs. [13] and [14] are based on the relationship between the average and maximum fitness values of the population. Similarly, the parameter F of DE in Ref. [33] is adaptively adjusted based on the minimum and maximum objective function values over the individuals in each generation. There are also some other examples found in the adaptive PSO algorithms [34–36].

3) Population distribution

Besides the fitness values, population distribution in the search space is another form of feedback for adaptation evidence. The population distribution is often used to reflect the population diversity. In such cases, all the population members are usually associated with the same adaptive parameters. A number of adaptive GAs control the crossover probability p_{x} or mutation probability p_{m} based on the population diversity. In these approaches, population diversity can be measured by

metrics such as Hamming distance [37,38] and other statistical information [39]. Another example is an adaptive DE algorithm (ADEA) [40] proposed for multiobjective optimization problems. In ADEA, the parameter F is decided by the number of Pareto-front and the relative crowding distance of the current solutions. Other AECs adjust the parameters for each individual based on its relative position. For example, the particles in some adaptive PSO adjust their own inertia weight or acceleration coefficients according to the relationship between their current positions and their personally best position [41] or the globally best position [42].

4) Combination of fitness value and population distribution

Some other adaptation evidences take both the fitness values and the population distribution information into consideration. In Refs. [11] and [12], such information is used to estimate the optimization states of the evolutionary process for GA and PSO, respectively. For the AGA [11], the population is first partitioned into clusters. Each cluster contains those individuals with similar component vectors. Then, the p_{x} and p_{m} are adjusted based on considering the relative size of cluster containing the best individual and the one containing the worst individual. For the APSO [12], the estimation of evolutionary state is based on the assumption that the mean distance from the globally best particle to other particles varies during the evolutionary process. In each generation, the relative mean distance of the globally best particle is calculated for evolutionary state estimation and the parameters are controlled accordingly. Different from the above two AECs, based on considering the fitness values and population distribution information, Nguyen et al. [43] proposed an adaptation mechanism for individual learning intensity and developed a probabilistic memetic framework. The characteristics and efficacies of this framework are demonstrated by both theoretical and empirical studies.

2.3 Adaptation methods

Designing a method for adaptation is the last step when we develop an adaptation scheme for EC. In this part we will introduce some commonly used adaptation methods like rule-based adaptation methods, coevolution methods, entropy-based control methods, and fuzzy control methods.

1) Simple rule-based adaptation methods

In many adaptation methods for EC, the adaptation is based on simple rules defined by the observations of runtime characteristics of the evolutionary algorithms. For example, in Ref. [44], the author proposed that the p_{m} in GA is changing in an exponential way. This method is proposed based on the observation on the convergence trends of GA algorithm. There are also many parameter adaptation methods based on the fitness values of each individual in the population. Different parameters are used for individuals according to their fitness values. In Ref. [13], the authors proposed the p_{x} and p_{m} adaptation rules based on the fitness information as follows (for minimization optimization problem):

$$p_{\mathrm{x}} = \begin{cases} p_{x_1} - \frac{(p_{x_1} - p_{x_2})(f'_t - f_{t,\mathrm{avg}})}{f_{t,\max} - f_{t,\mathrm{avg}}}, & \text{if } f'_t \geqslant f_{t,\mathrm{avg}}, \\ p_{x_1}, & \text{otherwise}, \end{cases} \tag{1}$$

where p_{x_1} and p_{x_2} are constant values which are used to control the range of the crossover probability. The motivation of Eq. (1) is that when the fitness value of an individual is better than the average fitness value, then the individual will use a large crossover rate and vice versa.

There are also similar approaches in PSO algorithms. In Ref. [35], the authors proposed a fitness based method to adjust the parameter w as

$$w = k_1 \left| \frac{f' - f_{\min}}{f_{\max} - f_{\min}} \right| + \varepsilon \cdot k_2, \tag{2}$$

where f' is the current fitness value of the particle, $f_{\min}$ is the current best fitness value of the whole population, and $f_{\max}$ is the optimal fitness of the worst particle at present. k_1 and k_2 are two constant factors used to control the range of the generated parameter, and ε is a random factor used to enhance the diversity of the generated parameter. With better fitness, the generated parameter will be smaller. In Ref. [45], the authors also used the fitness information to define rules to control the parameter for PSO, but in a more complicated form:

$$\omega_i^t = \omega_1 \cdot 2 \left(1 - \cos\left(\frac{\pi}{2}\xi_i^t\right)\right) + \omega_2, \tag{3}$$

where

$$\xi_i^t = \begin{cases} (f_i^{t-1} - f_G^{t-1})/f_i^{t-1}, & f_i^{t-1} \neq 0, \\ 0, & f_i^{t-1} = 0. \end{cases} \tag{4}$$

Different from the other fitness based methods using direct linear transformation, in the above method, the fitness information is normalized by a cosine function, which is nonlinear. Exponential transformation is also used to apply the fitness information to adjust the parameter w in PSO [41].

Covariance matrix adaptation ES (CMA-ES) [46–49] is one of the most promising evolutionary algorithms which can be used to solve some very difficult functions. The key mechanism of CMA-ES is the covariance matrix adaptation (CMA). CMA is based on the observation that the distribution model should be updated in an accumulated way. The evolution path information is also used to adapt the covariance matrix in order to let the algorithm explore along the current improving direction.

CMA-ES shows its efficiency on the complex functions due to the adaptation method.

2) Coevolution methods

Coevolution methods use some kind of parameter generation mechanisms and put these parameters into evolution process [50–52]. The parameters are co-evolved with the population in every generation. Usually, different parameters are generated for every individual. The parameters of the better individuals will survive in the evolution process and will be used to generate the new parameters.

The self-adaptive methods usually directly encode the parameters of the EC algorithms into the evolution procedure itself. The parameters are directly encoded into the original individuals and goes through the evolution procedure as genes of the population with the same methods as the original genes [32,53]. There are also different approaches like that in Refs. [54,55], where the parameters go through a different evolution procedure from the original population. In Ref. [54], the parameters are affected by a Gaussian random perturbation and in Ref. [55], the parameters of an ACO algorithm are self-adaptively adjusted by PSO. In Ref. [56], Zhan and Zhang applied the PSO learning strategy to the DE algorithm so as to evolve the DE parameters during the evolutionary process. In Ref. [57], the authors proposed a fuzzy controller based on a co-evolving GA. The fuzzy controller is going through a separate GA to adjust the fuzzy rules.

3) Entropy-based control methods

Entropy-based control methods are widely used in EC algorithms for parameter adaptation [58–65]. In the information theory, entropy is used to measure uncertainty with a random variable. In EC algorithms, the generation of new individuals often contains some random factors and the probability of generating a certain solution can be analyzed based on the population information. Therefore, entropy can be used to analyze the population characteristic of EC algorithms and the parameters and operators can be adjusted accordingly. The most common entropy-based control method is to compute the Shannon entropy according to the probabilities p_i of generating certain individuals:

$$S(t) = \sum_{i=1}^{n} p_i(t) \log p_i(t). \tag{5}$$

From the above formula we can see that when the probability of generating a certain solution becomes high, the entropy will become smaller. The maximum entropy $S_{\max}$ can be achieved when all the probabilities are equal. Thus this entropy value can be used as an indicator for population convergence. Usually the probabilities of generating different individuals can be achieved by simple analysis. For example, in the ACO [60], the probability of generating an individual in the current population is computed by the rate of pheromone on the solution:

$$p_i(t) = \frac{\tau_i(t)}{\sum_{m \in pop} \tau_m(t)}, \tag{6}$$

where *pop* is the current population.

From the above analysis we can see that entropy is just a measure of uncertainty of the current state based on some kinds of p_i. There are many different ways to evaluate the uncertainty of the probability distribution. For example in Ref. [66], the probability distribution of generating different individuals is first computed based on the pheromone trails. Then, the average value is calculated and the number of probability values larger than the average value is counted to be a state value to show the balance state. If the count number is small, the probability distribution is unbalanced and thus the population is more converged.

The entropy indicator can be used in many ways. For example, it can be used in some other controllers like the fuzzy controller [11]. Besides, it can also be used directly. In Ref. [60], the entropy information is used to adjust the evaporating rate of pheromone trails. The local pheromone evaporation rate is set to a small value at the early stage of convergence (when the entropy is large) and is set to a large value when the population becomes converged to avoid stagnation. This can be done easily with the state evaluation quantity variable(s). The state evaluation variable can be used as a condition to perform adjustment or not [41,67], and can also be used in linear transformation [60,68] and nonlinear transformation [69].

4) Fuzzy control methods

Fuzzy controllers have been widely used currently [11,12,34,70,71]. A fuzzy controller converts an input (e.g., certain information on the population) to an output (e.g., actual actions for parameter control or operator control) based on a set of fuzzy rules. In Ref. [57] the authors summarized the fuzzy controller models usually applied to EC algorithms. The fuzzy controller model is different from the other parameter adaptation methods for its well defined process stages and forms. A fuzzy controller often has three procedures in a row to do the control work. First the input signals or other information are mapped into the membership functions and truth values. Fuzzy logic theory defines the concept of fuzzy set, a kind of set which permits an element to partially belong to the set. The degree an element belongs to a fuzzy set is indicated by the membership function valued in real interval [0, 1]. The mapping by the membership function is the fuzzification process of the original input message. After the mapping some rules will be invoked

and the membership information can then be processed and the results of different rules will be combined. These rules are in the form of "IF THEN" statement. The truth value of each rule will be computed. The defuzzification process will then be applied to transform these truth values and control actions into the actual actions to do the control.

Using the fuzzy controller in EC algorithms is quite simple. The membership functions are used to evaluate the current state and then rules can then be proposed based on the state considered. Finally the rules will be used to control the parameters. The design of the membership function can be various according to the state considered. For example in Ref. [66], a cloud model is used to evaluate the current state. Various state evaluation methods can be applied as long as they can be adapted to the form of a fuzzy membership function.

After computing the membership functions, the control rules can also be designed with great freedom. In Ref. [66] the controller is used to decide whether to do an additional pheromone updating or not. In Ref. [11] the controller is used to increase or decrease the parameter values.

3 AEC based on evolutionary state

As reviewed in Section 2, a lot of AECs have been proposed. In this section, we pay more attention to the AECs based on the evolutionary state. Our motivations of introducing the evolutionary state into ECs are based on the following two considerations.

First, the population and the searching behavior of the EC algorithm exhibit different characteristics in different states of evolution. We take PSO as an example. Similarly to the investigations in Ref. [12], a time-varying 2-D sphere function $f(x-r) = (x_1 - r)^2 + (x_2 - r)^2$, $x_i \in [-10, 10]$, is employed to help illustrate the dynamic of particles distribution of PSO. The total generation number is set as 100, and the minimum of the function initialized at $(-5, -5)$ shifts to (5, 5) at the 50th generation. When a global version PSO with 100 particles optimizing function f, the particle distribution in different running phases is illustrated in Fig. 2 (refer to Fig. 1 of Ref. [12]).

It can be observed in Fig. 2(a) that all the particles distribute uniformly in the search space at the beginning of the searching process. Later all the particles fly toward the global optimum at $(-5, -5)$ as in Fig. 2(b). The whole swarm then gathers close around the best particle as in Fig. 2(c). In Fig. 2(d) when the global optimum of function f is shifted, the best particle quickly jumps out from the center of the swarm and guides the swarm toward the new optima as in Fig. 2(e). In Fig. 2(f) the PSO reaches a second convergence. From Fig. 2 we can see that the population distributions of PSO at

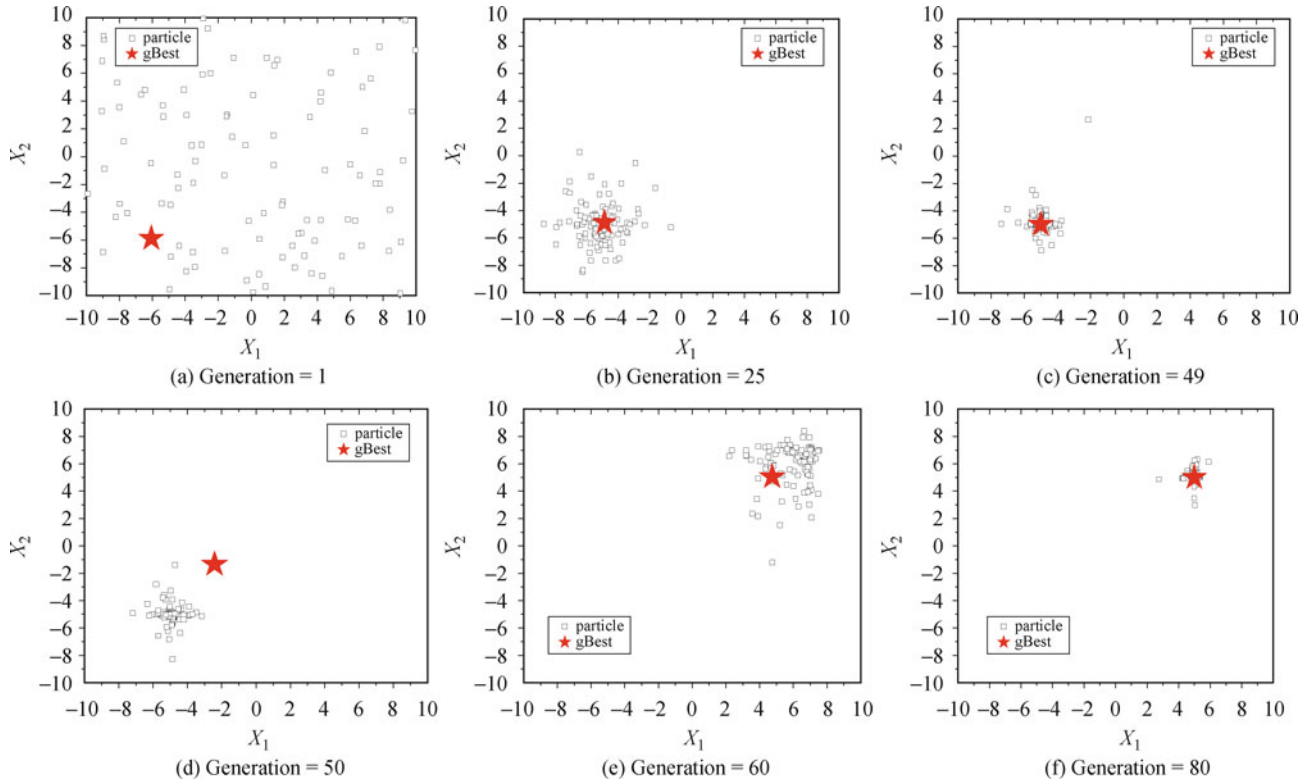


Fig. 2 Population distribution of PSO at different stages. (a) Generation = 1; (b) Generation = 25; (c) Generation = 49; (d) Generation = 50; (e) Generation = 60; (f) Generation = 80

different stages differ significantly from each other. The best particle is closely surrounded by the swarm when the algorithm has converged, whereas the best particle is far away from the crowding swarm when a new promising region is discovered. Otherwise, when the algorithm focuses on exploration, the whole swarm makes a comprehensive and scattered coverage of the search space.

The above observations suggest that the searching process of PSO can be identified as in one of several states. These states are termed as evolutionary state as defined in Ref. [12]. In each evolutionary state, the characteristics of population, e.g., the positional relationship between the best particle and other particles, are different from the other states.

Second, the ECs in different evolutionary state need different algorithm settings. We can still take the PSO as an example. At the state aiming at exploration, a parameter setting which helps the particles to learn more from its self experience is preferred. Thus the PSO can make a thorough search on as many optima as possible and avoid being cheated by local ones. However, when the best particle jumps out from the local optima, the parameter setting should emphasize more on the social cognition, which will guide the swarm toward the newly discovered promising region. In general, the parameter setting should follow the objective of ECs in different evolutionary states.

Based on the above considerations, it is beneficial to adaptively adjust the algorithm configurations of ECs based on the evolutionary state. When designing the adaptive algorithms based on evolutionary state, the two following problems should be first settled.

• How to determine the evolutionary state of the ECs?

• How to control the algorithm configurations of ECs based on the evolutionary state?

By solving the above-stated problems, an implementation of AECs based on evolutionary state can be given. The AGA in Ref. [11] and the APSO in Ref. [12] are two successful examples, both of which are designed through studies on population distribution characteristics and fitness information of the algorithm. The detailed descriptions of the two algorithms are presented as follows.

3.1 Clustering-based adaptive GA

The basic flow of the proposed AGA [11] is similar to that of traditional GA, except that the values of p_{x} and p_{m} are updated according to the proposed parameter adaptation scheme in each generation.

The parameter control in the proposed AGA mainly involves three tasks. One is to depict the distribution of population using the clustering technique. Another task is the design of a set of fuzzy rules for maturity state estimation and parameter adjustment. The last task is to identify the maturity state and control the parameters based on the results of clustering and the fuzzy rules.

1) Clustering of population

The K-means [72,73] algorithm is employed to depict the population distribution by partitioning the population into K different clusters. Among the K clusters, the cluster containing the best individual is denoted as G_{B}, and the cluster containing the worst individual is G_{W}. For generalization, the sizes of the G_{B} and G_{W} are normalized by the differences of $|G_{\max}|$ (the size of the largest cluster) and $|G_{\min}|$ (the size of the smallest cluster) as in Eqs. (7) and (8):

$$\hat{G}_{\mathrm{B}} = \frac{|G_{\mathrm{B}}| - |G_{\min}|}{|G_{\max}| - |G_{\min}|}, \tag{7}$$

$$\hat{G}_{\mathrm{W}} = \frac{|G_{\mathrm{W}}| - |G_{\min}|}{|G_{\max}| - |G_{\min}|}, \tag{8}$$

where $\hat{G}_{\mathrm{B}}$ and $\hat{G}_{\mathrm{W}}$ denote the relative size of G_{B} and G_{W}, respectively.

The relationship between individual fitness and positional distribution in the search space is outlined by the normalized sizes of G_{B} and G_{W} (i.e., $\hat{G}_{\mathrm{B}}$ and $\hat{G}_{\mathrm{W}}$). These results provide necessary information for the estimation of maturity state and parameter control.

2) Fuzzy rules for maturity state estimation and parameter control

Based on the results of clustering, the maturity state of search is estimated. A set of fuzzy rules are designed for the maturity state estimation and the adjustments of p_{x} and p_{m}.

For each maturity state, mainly three questions are taken into consideration in the design of rules. First, the need to encourage or discourage the reproduction of population outside the existing clusters is considered. This is largely related to the crossover probability p_{x}. Second, the need to encourage or discourage the reproduction of population within the existing clusters should be taken into consideration. This is largely related to the mutation probability p_{m}. Finally, the need to combine the above two guidelines is considered.

The rules for the adjustments of p_{x} and p_{m} are tabulated in Table 2 (Table II in Ref. [11]). These rules are explained as follows.

Rule 1 — matured state: The best individual is in a large cluster, and the worst individual is in a small cluster. In this situation, the values of p_{x} and p_{m} are reduced.

The optimization process is considered to be in the *matured state*. This situation implies that a large number of similar individuals are likely to swarm together around the best solution. Chances of reproducing better

Table 2 Fuzzy rule for the control of p_{x} and p_{m}

rule for $\theta_{p_{\mathrm{x}}}$
Rule 1 = (Rule (1,0)): *if* ($\hat{G}_{\mathrm{B}}$ is PB) *and* ($\hat{G}_{\mathrm{W}}$ is PS) *then* $\theta_{p_{\mathrm{x}}}$ = NB
Rule 2 = (Rule (1,1)): *if* ($\hat{G}_{\mathrm{B}}$ is PB) *and* ($\hat{G}_{\mathrm{W}}$ is PB) *then* $\theta_{p_{\mathrm{x}}}$ = PB
Rule 3 = (Rule (0,0)): *if* ($\hat{G}_{\mathrm{B}}$ is PS) *and* ($\hat{G}_{\mathrm{W}}$ is PS) *then* $\theta_{p_{\mathrm{x}}}$ = PB
Rule 4 = (Rule (0,1)): *if* ($\hat{G}_{\mathrm{B}}$ is PS) *and* ($\hat{G}_{\mathrm{W}}$ is PB) *then* $\theta_{p_{\mathrm{x}}}$ = NB
rule for $\theta_{p_{\mathrm{m}}}$
Rule 1 = (Rule (1,0)): *if* ($\hat{G}_{\mathrm{B}}$ is PB) *and* ($\hat{G}_{\mathrm{W}}$ is PS) *then* $\theta_{p_{\mathrm{m}}}$ = NB
Rule 2 = (Rule (1,1)): *if* ($\hat{G}_{\mathrm{B}}$ is PB) *and* ($\hat{G}_{\mathrm{W}}$ is PB) *then* $\theta_{p_{\mathrm{m}}}$ = NB
Rule 3 = (Rule (0,0)): *if* ($\hat{G}_{\mathrm{B}}$ is PS) *and* ($\hat{G}_{\mathrm{W}}$ is PS) *then* $\theta_{p_{\mathrm{m}}}$ = PB
Rule 4 = (Rule (0,1)): *if* ($\hat{G}_{\mathrm{B}}$ is PS) *and* ($\hat{G}_{\mathrm{W}}$ is PB) *then* $\theta_{p_{\mathrm{m}}}$ = PB

individuals through crossover and mutation are relatively small. Thus the values of p_{x} and p_{m} are reduced.

Rule 2 — maturing state: The best individual is in a large cluster, and the worst individual is in a large cluster. In this situation, the value of p_{x} is increased and the value of p_{m} is reduced.

The optimization process is considered to be *maturing*. Here both G_{B} and G_{W} are likely to be the largest cluster. The algorithm has to achieve two goals. First, since the direction for exploration is largely undetermined, new search directions should be explored. Second, the convergence of population should be encouraged. To achieve the above two goals, the algorithm is expected to reproduce more individuals in and around the existing clusters, and to avoid breeding individuals outside the current population. Thus the value of p_{x} is increased and the value of p_{m} is reduced.

Rule 3 — sub-maturing state: The best individual is in a small cluster, and the worst individual is in a small cluster. In this situation, the values of p_{x} and p_{m} are increased.

The optimization process is considered to be in the *sub-maturing state*. Both G_{B} and G_{W} are probably the smallest cluster. The region containing global optimum has not been located and the direction of search is undetermined. To explore new search directions, breeding more individuals outside the existing population is required. On the other hand, the population has not swarmed around the best solution. The algorithm is expected to reproduce more offspring in and around G_{B} for local exploitation. Thus the values of p_{x} and p_{m} are increased.

Rule 4 — initial state: The best individual is in a small cluster, and the worst individual is in a large cluster. In this situation, the value of p_{x} is reduced and the value of p_{m} is increased.

The optimization process is considered to be in the *initial state*. The situation implies that the evolution has just started. In the initial stage of evolution, finding new search directions is necessary for the exploration of the whole search space. Thus the value of p_{m} is increased and the value of p_{x} is decreased.

3) Fuzzy control of p_{x} and p_{m}

The mechanism of adjusting the values of p_{x} and p_{m} is based on fuzzy control, which consists of three components, i.e., fuzzification of variables, rule-based fuzzy inference, and defuzzification.

Fuzzification of variables: Two input variables, i.e., $\hat{G}_{\mathrm{B}}$ and $\hat{G}_{\mathrm{W}}$, are mapped to linguistic values through fuzzy membership functions. For each of the variables, two fuzzy subsets are defined, including *positive small* (PS) and *positive big* (PB). The membership functions for the two fuzzy subsets on each variable are given in Eqs. (9) and (10):

$$\mu_0(\hat{G}) = 1 - \hat{G}, \tag{9}$$

$$\mu_1(\hat{G}) = \hat{G}. \tag{10}$$

Rule-based fuzzy inference: Based on the fuzzified input variables and the fuzzy rules in Table 2, the control actions of parameters p_{x} and p_{m} are inferred. The fuzzy inference considers all the four rules. Take Rule (1,0) for example, the membership of the output fuzzy set $m_{1,0}$ is calculated by multiplying $\mu_1(\hat{G}_{\mathrm{B}})$ and $\mu_0(\hat{G}_{\mathrm{W}})$, where

$$m_{1,0} = \mu_1(\hat{G}_{\mathrm{B}})\mu_0(\hat{G}_{\mathrm{W}}). \tag{11}$$

Defuzzification: To output the changes of the values of p_{x} and p_{m}, the results of fuzzy inference should be converted to crisp values. The actual values of p_{x} and p_{m} are the sum of their values in the previous generation ($p_{\mathrm{x},gen-1}$ and $p_{\mathrm{m},gen-1}$) and the output of the fuzzy controller, as given in Eqs. (12) and (13):

$$p_{\mathrm{x},gen} = p_{\mathrm{x},gen-1} + K_{\mathrm{x}}\theta_{p_{\mathrm{x}}}, \tag{12}$$

$$p_{\mathrm{m},gen} = p_{\mathrm{m},gen-1} + K_{\mathrm{m}}\theta_{p_{\mathrm{m}}}. \tag{13}$$

Here K_{x} and K_{m} are used to control the changes of p_{x} and p_{m} to be within a reasonable level. The variables $\theta_{p_{\mathrm{x}}}$ and $\theta_{p_{\mathrm{m}}}$ are the defuzzified fuzzy decisions derived from the four fuzzy rules. The defuzzification is applied according to the formulas (14) and (15):

$$\theta_{p_{\mathrm{x}}} = \frac{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1} m_{1,0}y_{ij}}{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1} m_{1,0}}, \tag{14}$$

$$\theta_{p_{\mathrm{m}}} = \frac{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1} m_{1,0} z_{ij}}{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1} m_{1,0}}, \tag{15}$$

where the values of y_{ij} and z_{ij} are either $+1$ or -1, which are determined by the corresponding rule. For example, the action in Rule (1,0) for $\theta_{p_{\mathrm{x}}}$ is "NB", and the value of y_{10} is -1.

4) Experimental results

In Ref. [11], experiments are conducted on two groups of problems to investigate the performance of the proposed AGA. The first group includes a set of mathematical functions, whereas the second group includes an application problem of PEC.

On all of the mathematical functions, the proposed AGA with controlled parameters consumes a smaller number of evaluations to achieve the global optimum than traditional GA. On the application problem, the proposed method significantly improves the performance of GA in terms of search speed and solution quality. These results verified the effectiveness of the proposed parameter control strategy.

3.2 Adaptive PSO

The basic idea of APSO is to adaptively control the parameter and strategy of PSO according to the current state revealed in the evolutionary process of PSO. The adaptation procedure of APSO consist mainly two steps, the evolutionary state estimation (ESE) step and the parameter and strategy control (PSC) step.

• ESE step: evaluate the current evolutionary state of APSO based on the population distribution and particle fitness.

• PSC step: adaptively control the parameter and strategy of APSO according to the current evolutionary state.

1) Evolutionary state estimation

Zhan et al. [12] have proposed a novel ESE approach by analyzing the search behavior and the population distribution characteristic of the PSO. In Ref. [12], four evolutionary states of PSO have been presented, namely *exploration*, *exploitation*, *convergence*, and *jumping-out* states. The authors indicated that the positional relationship between the best particle and the other particles is an efficacious measure to mark off the current evolutionary state of PSO as one of the four states. When the best particle is surrounded by the swarm, the PSO is more likely to be in the *convergence* state. At this point, the globally best particle obtains a minimal mean distance to the whole swarm than any other particle. Otherwise, when the best particle is far away from the crowding swarm, the PSO is more likely to be in the *jumping-out* state. At this point, the globally best particle obtains a maximal mean distance to the whole swarm. Hence, a normalized evolutionary factor f reflecting the comparatively distance from the best particle to the whole swarm is defined as Eq. (16) to indicate the current state of PSO:

$$f = \frac{d_{\mathrm{gbest}} - d_{\min}}{d_{\max} - d_{\min}}, \tag{16}$$

where d_i is the mean Euclidian metric between the ith particle and the whole swarm which is formulized as

$$d_i = \frac{1}{N-1}\sum_{j=1}^{N}\sqrt{\sum_{k=1}^{D}(x_i^k - x_j^k)}, \tag{17}$$

N and D are the population size and the problem dimension respectively. Thus d_{gbest} is the mean Euclidian metric between the globally best particle and the whole swarm, and $d_{\max}$ and $d_{\min}$ are the maximum and minimum distances of all d_i's.

A fuzzy classifier is adopted to map the normalized evolutionary factor f into the current evolutionary state of PSO as in Fig. 3.

2) Parameter and strategy control

a) Adaptive parameter control

Three parameters, i.e., the inertia weight w, and the acceleration coefficients c_1 and c_2 are adaptively

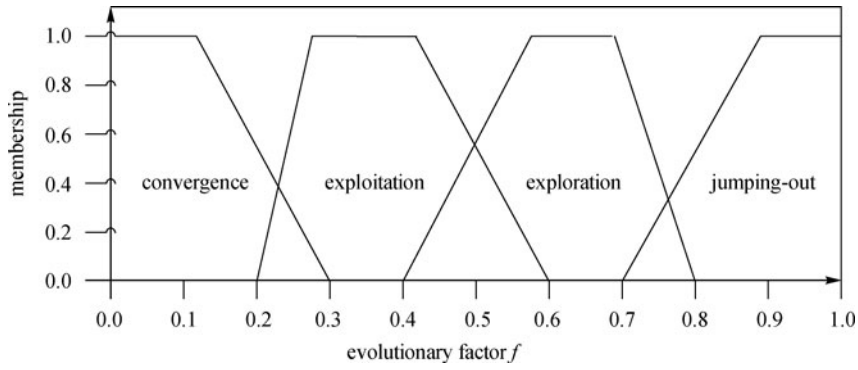


Fig. 3 Fuzzy membership functions for the four evolutionary states

adjusted based on the normalized evolutionary factor f and current evolutionary state obtained in the ESE steps. The details are as follows.

The inertia weight w is an important parameter which places a vital influence on balancing the exploration and exploitation ability of PSO. A comparatively larger w is preferred for PSO to make a more thoroughly search in the exploration state and a comparatively smaller w is preferred to accelerate the converging speed of PSO in the exploitation state. The adaptation of the inertia weight w is illustrated as

$$w(f) = \frac{1}{1 + 1.5\mathrm{e}^{-2.6f}} \in [0.4, 0.9], \tag{18}$$

where f is the normalized evolutionary factor computed as Eq. (16).

The acceleration coefficients c_1 and c_2 are two parameters deciding the relative weight that the particle learns from its own experience, i.e., the self-cognition, or the whole swarm's experience, i.e., the social influence. PSO with a larger weight of c_1 is more likely to pull the particle towards its historical best position, thus keeps the diversity and leads to a more thorough exploration of all the local niches. On the contrary, PSO with a larger weight of c_2 is more likely to push the swarm towards the global best region, thus enjoys a fast converging speed. Four operations, i.e., increase (++), slightly increase (+), decrease (– –) and slightly decrease (–) are employed on acceleration coefficients c_1 and c_2 according to the evolutionary state. The operation increase (++) or slightly increase (+) on c_i means that one "acceleration rate" δ or half of δ is added to c_i, respectively. The decrease operations are analogous with the increase operations. The adaptations of the acceleration coefficients c_1 and c_2 are detailed as shown in Table 3.

Table 3 Fuzzy control of the acceleration coefficients c_1 and c_2

evolutionary state	adjustment on c_1 and c_2
jumping-out	– – c_1 and ++ c_2
exploration	++ c_1 and – – c_2
exploitation	+ c_1 and – c_2
convergence	+ c_1 and + c_2

Jumping-out state: Decrease c_1 and increase c_2. At this state, the global best particle has jumped out of the crowding swarms and obtained a better solution. By decreasing c_1 and increasing c_2, much more emphasis is put on the social experience in the learning process of the particles, which can help the whole swarm to move to the promising region as quickly as possible.

Exploration state: Increase c_1 and decrease c_2. At the exploration state, a thorough search on as many optima as possible is expected. Increasing the weight of c_1 forces the particles to fly close around their historical best position and explore individually in a variety of promising regions, instead of gathering together around the current best particle.

Exploitation state: Slightly increase c_1 and slightly decrease c_2. At the exploitation state, the particles are grouping into different clusters around the corresponding local optima of their historical best positions. A full exploitation of all the potential local niches is expected at this state to avoid the deception of the local optima. Slightly increasing the weight of c_1 can achieve this target.

Convergence state: Slightly increase c_1 and slightly increase c_2. At this state, a global optima region seems to be found. Slightly increasing c_2 can attract the other particles to fly toward the probable optima region. On the other hand, by slightly increasing c_1, the PSO can avoid too fast a converging speed which may lead to over-premature.

After the adaptive adjustment on the acceleration coefficients c_1 and c_2, a bounding and normalization operation is performed on them to ensure a more stable performance of APSO. Both c_1 and c_2 are clamped between [1.5, 2.5], and their sum is bounded between [3.0, 4.0]. If the sum of the two parameters exceeds the upper bound 4.0, c_1 and c_2 are normalized to

$$c_i = \frac{c_i}{c_1 + c_2} \times 4.0. \tag{19}$$

Note that after the bounding and normalization operation of c_1 and c_2, slightly increasing c_1 and slightly increasing c_2 in the convergence state will eventually lead to a slight decrease on c_1 and a slight increase on c_2 because c_1 and c_2 will be drawn to 2.0 when they are both beyond the upper bound 4.0.

b) Adaptive strategy control

As the global best particle of PSO has no exemplars to follow, Ref. [12] proposed a perturbation-based elitist learning strategy (ELS) to give some fresh momentum to the global leader. The ELS randomly chooses one dimension of the best particle's historical best position and performs a Gaussian perturbation as follows:

$$P^d = P^d + (X^d_{\max} - X^d_{\min}) \times Gaussian(\mu, \delta^2), \tag{20}$$

where $[X^d_{\min}, X^d_{\max}]$ is the search bound of the problem. $Gaussian(\mu, \delta^2)$ is a random number following the Gaussian distribution. The mean μ of the Gaussian distribution is set as zero and the standard deviation δ is linearly decreasing with the evolutionary generation as

$$\delta = \partial_{\max} - (\partial_{\max} - \partial_{\min}) \times \frac{g}{G}, \tag{21}$$

where g and G are the current generation and total generations respectively. $\partial_{\max}$ and $\partial_{\min}$ are the upper and lower bound of ∂ which are empirically set as 1 and 0.1.

The employment of different strategies is also adaptively adjusted according to the current evolutionary state of PSO. The ELS is employed by the APSO only in

the convergence state to help the best particle to jump out of the probable local optima regions and enhance the globally search ability of APSO. Nevertheless, in the other states, the classical PSO learning strategy is employed because the problem of jumping out from the local optima is not the critical issue in these states.

4 Potential research directions

Although during the last decades, research on the adaptation of ECs has reached an impressive state, there are still many open problems and exciting issues to be researched. Below, we unfold some important future directions in the area of EC algorithm adaptation.

First, the relationship between the algorithm adaptation (especially the parameters adaptation) and the performance of ECs should be further investigated. A more clear clarification on the mechanism of how the parameters work should presented. This is very important to design parameter adaptation methods. In addition, as there are strong similarities and connections between different EC algorithms, it is very interesting to study whether there is a common parameter adaptation framework for all the EC algorithms.

Second, more researches should be carried out on the theoretical analysis on the algorithm adaptation of ECs. Most techniques used to adapt the algorithm configurations in ECs are simple and based on the experience of the designers. The theoretical studies can provide a solid foundation and a powerful guidance in designing the adaptation strategies for ECs. Moreover, whether the ECs can still preserve the convergence characteristic by using algorithm adaptation is worth of studying.

Third, the techniques from machine learning (ML) and other domains should be introduced to the EC algorithm adaptation control methodology. The ML techniques, which have attracted a lot of interests from researchers all around the world, own firm theoretical foundation and show favorable performance on many applications. Recent researches have also shown that incorporating ML techniques into EC algorithms can improve their performance significantly [54,74]. For example, inspired from the idea of ensemble learning in ML domain, Qin et al. [54] proposed a similar concept for DE to form an ensemble of mutation strategies and parameter values which competes to produce successful offspring population. In addition, reinforcement learning can be used to design a new parameter reinforcement learning strategy which could benefit from the profound reinforcement learning theory. Theory of automatic control should be further used in current AECs.

Fourth, the algorithm adaptation methodology of ECs in complex environments should be further studied. Currently, the researches on AECs are mainly focused on normal discrete and continuous optimization problems. However, a lot of complex optimization, e.g., multiobjective optimization, constrained optimization, large-scale optimization, dynamic and uncertain environment, has drawn more and more attention from researchers. The algorithm adaptation schemes which exhibit favorable performance on normal optimization problems may act poorly on complex optimization problems. Hence, how to design promising algorithm adaptation schemes based on the special demands of these complex optimization problems is very meaningful and still leaves much to be done.

5 Conclusion

In this paper, we have provided a comprehensive survey on adaptation schemes in ECs. Following the steps of designing an adaptation approach, the taxonomy of work in this area has been defined. The proposed classification is based on three aspects: 1) adaptation objects, 2) adaptation evidences, and 3) adaptation methods. Adaptation schemes found in the literature have been classified into corresponding categories and extensively discussed. Such a survey provides an overall picture of the relevant research in the area and can be served as a guide when designing an adaptation scheme for ECs. Moreover, a kind of promising adaptation methods based on estimation of evolutionary state have been further discussed. Two representative AECs using such adaptation method are introduced in detail. In the end of this paper, some potential research directions have also been suggested based on the survey of the existing approaches.

Acknowledgements This work was supported in part by the National Science Fund for Distinguished Young Scholars (No. 61125205), the National Natural Science Foundation of China (NSFC) (Grant No. 61070004), and NSFC Joint Fund with Guangdong under Key Project U0835002.

References

1. Eiben A E, Smith J E. Introduction to Evolutionary Computation. Springer, 2003
2. De Jong K A. An analysis of the behaviour of a class of genetic adaptive systems. Dissertation for the Doctoral Degree. Ann Arbor, MI: University of Michigan, 1975
3. Bäck T, Hammel U, Schwefel H. Evolutionary computation: Comments on the history and current state. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 3–17
4. Hesser J, Männer R. Towards an optimal mutation probability for genetic algorithms. Lecture Notes in Computer Science, 1991, 496: 23–32
5. Schwefel H P. Numerical Optimisation of Computer Models. New York: Wiley, 1981

6. Bäck T, Fogel D B, Michalewicz Z. Handbook of Evolutionary Computation. Bristol: Institute of Physics Publishing and New York: Oxford University Press, 1997
7. Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison Wesley, 1989
8. Stephens C R, García Olmedo I, Mora Vargas J, Waelbroeck H. Self-adaptation in evolving systems. Artificial Life, 1998, 4(2): 183–201
9. Zhan Z H, Xiao J, Zhang J, Chen W. Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis. In: Proceedings of IEEE Congress on Evolutionary Computation. 2007, 3276–3282
10. Eiben A E, Hinterding R, Michalewicz Z. Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 124–141
11. Zhang J, Chung H S H, Lo W L. Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Transactions on Evolutionary Computation, 2007, 11(3): 326–335
12. Zhan Z H, Zhang J, Li Y, Chung H S H. Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(6): 1362–1381
13. Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 1994, 24(4): 656–667
14. Dai C H, Zhu Y F, Chen W R. Adaptive probabilities of crossover and mutation in genetic algorithms based on cloud model. In: Proceedings of IEEE Information Theory Workshop. 2006, 710–713
15. Shi Y, Eberhart R. A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation. 1998, 69–73
16. Cai Z, Huang H, Qin Y, Ma X. Ant colony optimization based on adaptive volatility rate of pheromone trail. International Journal of Communications, Network and System Sciences, 2009, (2): 792–796
17. Hao Z, Huang H, Qin Y, Cai R. An ACO algorithm with adaptive volatility rate of pheromone trail. In: Proceedings of International Conference of Computational Science. 2007, 1167–1170
18. Mezura-Montes E, Velázquez-Reyes J, Coello C A C. A comparative study of differential evolution variants for global optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. 2006, 485–492
19. Shi X H, Wan L M, Lee H P, Yang X W, Wang L M, Liang Y C. An improved genetic algorithm with variable population size and a PSO-GA based hybrid evolutionary algorithm. In: Proceedings of the 2nd International Conference on Machine Learning and Cybernetics. 2004, 1735–1740
20. Tse S M, Liang Y, Leung K S, Lee K H, Mok T S K. A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2007, 37(1): 84–91
21. Teng N S, Teo J, Hijazi M H A. Self-adaptive population sizing for a tune-free differential evolution. Soft Computing, 2009, 13(7): 709–724
22. Teo J. Exploring dynamic self-adaptive populations in differential evolution. Soft Computing, 2006, 10(8): 673–686
23. Tan K C, Lee T H, Khor E F. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. IEEE Transactions on Evolutionary Computation, 2001, 5(6): 565–588
24. Kennedy J, Mendes R. Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation. 2002, 1671–1676
25. Janson S, Middendorf M. A hierarchical particle swarm optimizer and its adaptive variant. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2005, 35(6): 1272–1282
26. Montes de Oca M A, Stutzle T, Birattari M, Dorigo M. Frankenstein's PSO a composite particle swarm optimization algorithm. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 1120–1132
27. Ong Y S, Lim M H, Zhu N, Wong K W. Classification of adaptive memetic algorithms: A comparative study. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2006, 36(1): 141–152
28. Zhang J, Chung H S H, Lo W L. Pseudocoevolutionary genetic algorithms for power electronic circuits optimization. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2006, 36(4): 590–598
29. Zhan Z H, Zhang J. Parallel particle swarm optimization with adaptive asynchronous migration strategy. Lecture Notes in Computer Science, 2009, 5574: 490–501
30. Das S, Konar A, Chakraborty U K. Two improved differential evolution schemes for faster global search. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. 2005, 991–998
31. Merkle D, Middendorf M, Schmeck H. Ant colony optimization for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 333–346
32. Zhang J, Sanderson A C. JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945–958
33. Ali M M, Törn A. Population set based global optimization algorithms: Some modifications and numerical studies. Computers & Operations Research, 2004, 31(10): 1703–1725
34. Shi Y, Eberhart R C. Fuzzy adaptive particle swarm optimization. In: Proceedings of the 2001 Congress on Evolutionary Computation. 2001, 101–106
35. Zhu J, Zhao J, Li X. A new adaptive particle swarm optimization algorithm. In: Proceedings of the 2008 International Workshop on Modelling, Simulation and Optimization. 2008, 456–458
36. Pappala V S, Erlich I. A new approach for solving the unit commitment problem by adaptive particle swarm optimization. In: Proceedings of Power and Energy Society General Meeting. 2008, 1–6
37. Liu X P, Liu Y. Adaptive genetic algorithm based on population diversity. In: Proceedings of International Forum on

Information Technology and Applications. 2009, 510–512

38. Zhu K Q. A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. In: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence. 2003, 176–183
39. Wang H F, Yang S X, Ip W H, Wang D W. Adaptive primal-dual genetic algorithms in dynamic environments. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(6): 1348–1361
40. Qian W Y, Li A. Adaptive differential evolution algorithm for multiobjective optimization problems. Applied Mathematics and Computation, 2008, 201(1–2): 431–440
41. Li X, Fu H, Zhang C. A self-adaptive particle swarm optimization algorithm. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering. 2008, 186–189
42. Wang L, Kang Q, Xiao H, Wu Q D. A modified adaptive particle swarm optimization algorithm. In: Proceedings of IEEE International Conference on Industrial Technology. 2005, 209–214
43. Nguyen Q H, Ong Y S, Lim M H. A probabilistic memetic framework. IEEE Transactions on Evolutionary Computation, 2009, 13(3): 604–623
44. Fogarty T C. Varying the probability of mutation in the genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms. 1989, 104–109
45. Wu Z, Zhou J. A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment. In: Proceedings of the 2007 International Conference on Computational Intelligence and Security. 2007, 133–136
46. Auger A, Hansen N. A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 Congress on Evolutionary Computation. 2005, 1769–1776
47. Igel C, Hansen N, Roth S. Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation, 2007, 15(1): 1–28
48. Shir O M, Bäck T. Niche radius adaptation in the CMA-ES niching algorithm. Lecture Notes in Computer Science, 2006, 4193: 142–151
49. Suttorp T, Hansen N, Igel C. Efficient covariance matrix update for variable metric evolution strategies. Machine Learning, 2009, 75(2): 167–197
50. Spears W M. Adapting crossover in evolutionary algorithms. In: Proceedings of the 4th Annual Conference on Evolutionary Programming. 1995
51. Angeline P J, Fogel D B, Fogel L J. A comparison of self-adaptation methods for finite state machines in dynamic environments. In: Proceedings of the 5th Annual Conference on Evolutionary Programming. 1996
52. Bäck T. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: Proceedings of the 2nd Conference on Parallel Problem Solving from Nature. 1992, 85–94
53. Hinterding R. Gaussian mutation and self-adaption in numeric genetic algorithms. In: Proceedings of the 2nd IEEE Conference on Evolutionary Computation. 1995, 384–389
54. Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398–417
55. Hao Z F, Cai R C, Huang H. An adaptive parameter control strategy for ACO. In: Proceedings of the 2006 International Conference on Machine Learning and Cybernetics. 2006, 203–206
56. Zhan Z H, Zhang J. Self-adaptive differential evolution based on PSO learning strategy. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. 2010, 39–46
57. Herrera F, Lozano M. Adaptive genetic operators based on coevolution with fuzzy behaviors. IEEE Transactions on Evolutionary Computation, 2001, 5(2): 149–165
58. Yang L, Widyantoro D H, Ioerger T, Yen J. An entropy-based adaptive genetic algorithm for learning classification rules. In: Proceedings of the 2001 Congress on Evolutionary Computation. 2001, 790–796
59. Xing Y, Chen Z, Sun J, Hu L. An improved adaptive genetic algorithm for job-shop scheduling problem. In: Proceedings of the 3rd International Conference on Natural Computation. 2007, 287–291
60. Li Y, Zhou S. Adaptive ACO for complicated optimization problems. In: Proceedings of the 2009 International Forum on Information Technology and Applications. 2009, 15–18
61. Ahmadi P, Hajabdollahi H, Dincer I. Cost and entropy generation minimization of a cross-flow plate fin heat exchanger using multi-objective genetic algorithm. Journal of Heat Transfer — Transactions of the ASME, 2011, 133(2): 021801
62. Fu P, Luo K. Clustering analysis of immune-genetic algorithm based on information entropy. Computer Engineering, 2008, 34(6): 227–232 (in Chinese)
63. Wang J, Liu B. Advanced genetic algorithm based two-dimensional fuzzy entropy image segmentation algorithm. Science & Technology Review, 2010, 28(20): 43–47
64. Xue F, Wang C-g, Mu F. Genetic and ant colony collaborative optimization algorithm based on information entropy and chaos theory. Control and Decision, 2011, 26(1): 44–48 (in Chinese)
65. Shen Z, Hao Y, Li K. Application research of an adaptive genetic algorithms based on information entropy in path planning. In: Proceedings of the 2010 International Conference on Information and Automation. 2010
66. Li Z, Wang Y, Yu J, Zhang Y, Li X. A novel cloud-based fuzzy self-adaptive ant colony system. In: Proceedings of the 4th International Conference on Natural Computation. 2008, 460–465
67. Xiong W, Wang C. Feature selection: A hybrid approach based on self-adaptive ant colony and support vector machine. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering. 2008, 751–754
68. Yu W J, Hu X M, Zhang J, Huang R. Self-adaptive ant colony system for the traveling salesman problem. In:

Proceedings of IEEE International Conference on Systems, Man and Cybernetics. 2009, 1399–1404

69. Favuzza S, Graditi G, Sanseverino E R. Adaptive and dynamic ant colony search algorithm for optimal distribution systems reinforcement strategy. Applied Intelligence, 2006, 24(1): 31–42
70. Wu H, Shi Y F, Jin X, Wang G, Dong H. A fuzzy adaptive particle swarm optimization for RNA secondary structure prediction. In: Proceedings of the 2011 International Conference on Information Science and Technology. 2011, 1390–1393
71. Chang W, Luo X, Yu H. A fuzzy adaptive particle swarm optimization for long-term optimal scheduling of cascaded hydropower station. In: Proceedings of IEEE/PES Power Systems Conference and Exposition. 2009, 1–5
72. Tou J T, Gonzalez R C. Pattern Recognition Principles. Reading, MA: Addison-Wesley, 1974
73. Duda R O, Hart P E, Stork D G. Pattern Classification. New York: Wiley, 2001
74. Zhang J, Zhan Z H, Lin Y, Chen N, Gong Y J, Zhong J H, Chung H S H, Li Y, Shi Y H. Evolutionary computation meets machine learning: A survey. IEEE Computational Intelligence Magazine, 2011, 6(4): 68–75

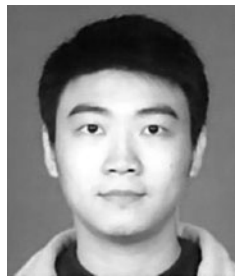
Jun ZHANG (M'02–SM'08) received the Ph.D. degree from Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China, in 2002.

He is currently a Professor with the Department of Computer Science, Sun Yat-sen University, Guangzhou, China. He has authored seven research books and book chapters, and over 100 technical papers in his research areas. His research interests include computational intelligence, operations research, data mining, wireless sensor networks, and power electronic circuits.

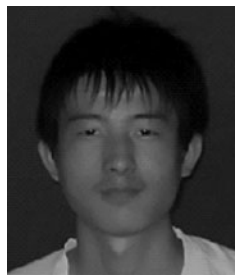
Dr. Zhang received the National Science Fund for Distinguished Young Scholars from the National Natural Science Foundation of China in 2011. He received the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. Dr. Zhang is currently an Associate Editor for the IEEE Transactions on Industrial Electronics, and Associate Editor for IEEE Computational Intelligence Magazine. He is the founding and current Chair of the IEEE Guangzhou Subsection and the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapter.

Wei-Neng CHEN received the Bachelor's degree in network engineering in 2006 from the Sun Yat-sen University, Guangzhou, China, where he is currently working toward the Ph.D. degree in the Department of Computer Science. His current research interests include evolutionary computation and its applications on financial optimization and operations research.

Zhi-Hui ZHAN received the Bachelor's degree in computer science and technology from Sun Yat-sen University, Guangzhou, China, in 2007, where he is currently working toward the Ph.D. degree. His current research interests include particle swarm optimization, ant colony optimization, differential evolution, genetic algorithms, and their applications in real-world problems.

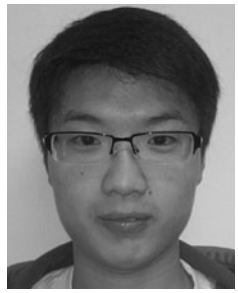


Wei-Jie YU received his Bachelor's degree in network engineering in 2009 from Sun Yat-sen University, Guangzhou, China, where he is currently working toward his Ph.D. degree in the Department of Computer Science. His current research interests include differential evolution, ant colony optimization, particle swarm optimization, and other evolutionary algorithms.



Yuan-Long LI received the B.A. degree in mathematics from Sun Yat-sen University, Guangzhou, China, in 2009. He is currently working toward the Ph.D. degree in computer science from the Sun Yat-sen University, Guangzhou, China. Currently, his main research areas are meta-heuristics and evolutionary computation, which includes differential evolution, CMA-ES, and so on.

Ni CHEN received the Bachelor's degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2009, where she is currently working toward the Ph.D. degree with the Department of Computer Science. Her current research interests include particle swarm optimization, genetic algorithms, and other computational intelligence techniques.



Qi ZHOU received the B.S. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2010. Now, he is working toward the Ph.D. degree. His current research interests include artificial intelligence, evolutionary computation, swarm intelligence.