Enhancing the Performance of Evolutionary Algorithms: a Novel Maturity-Based Adaptation Strategy

Yu Guo, Wei-Neng Chen, and Jun Zhang (Corresponding Author)
Sun Yat-sen University, P.R. China
Key Laboratory of Digital Life, Ministry of Education, P.R. China
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

***Abstract*—Adapting genetic operators and parameter settings during the optimization process can improve the overall performance of evolutionary algorithms (EAs). In this paper, a novel maturity-based adaptation strategy for EAs is proposed. During the search process, a maturity degree of the population is calculated based on both the population distribution in the search space and that in the fitness space. According to the maturity degree, four evolution states of EAs are defined by a set of thresholds. As both the convergence of the genotypes (reflected by the geographical distance among the individuals) and that of the phenotypes (shown by the differentiation over individuals' fitness values) are taken into consideration, the estimation of the evolution state is very comprehensive. Then, a set of adaptation rules is applied to adapt the parameters and operators of EAs according to the maturity degree and evolution state. Implemented on genetic algorithm (GA), the probabilities of crossover and mutation are tuned to fulfill the current evolution requirement of the population. Meanwhile, a novel allele gene-based mutation scheme and the traditional mutation are alternately executed. Experimental results on eight benchmark functions show that the proposed maturity-based adaptation strategy can bring significant improvements in search speed, solution accuracy and robustness.**

Keywords-Adaptation strategy, evolutionary algorithm (EA), genetic algorithm (GA)

I. INTRODUCTION

In recent years, evolutionary algorithms (EAs) [1][2] have gained significant attention due to their clear concepts, easy implementations and generality in problem solving, robustness, etc. However, current EAs still encounter some problems, e.g., they may be parameter sensitive, be vulnerable to local optima, and have slow convergence. For example, although the genetic algorithms (GAs) [3][4] are robust for not requiring domain knowledge of optimization problems, they are quite sensitive to the control parameters. The optimal settings of the crossover probability (p_x) and mutation probability (p_m) in GAs are usually problem dependent. Moreover, even in a same problem, the required settings of p_x and p_m differ within different evolution stages. Also, the convergence speed of GA needs to be accelerated by adapting the proper operator in certain evolution stages. Using fixed values of p_x and p_m and the unaltered operator cannot fully exploit the search effectiveness and efficiency of GAs.

The adaptation for the parameters [5] and operator during the search process of EAs is thus become a crucial issue. In the literature, different kinds of adaptation techniques of EAs have been proposed and developed. Take GA as an example, M. Srinivas and L. M. Patnaik first proposed the adaptive GA (AGA) [6], which adopted a deterministic adaptation function based on the best and the average fitness values in each generation. In [7], the performance of AGA was further improved by introducing cloud model and randomness. An allele gene-based adaptive GA [8] introduced a normalized resemblance factor (RF) and an allele gene diversity (AGD) to characterize the population diversity in order to adjust the p_m. In [9], a clustering-based adaptive GA was developed, which applied clustering analysis to identify the current evolution stage and then adjusted the values of p_x and p_m accordingly. The above adaptation strategies can be categorized into three groups. In the first group [6][7], the adjustment of parameters is based on the fitness information of the current population. The works in [8] belong to the second group. Instead of considering the fitness values, they adjusted the parameter settings according to the population distribution in the search space. The clustering-based adaptation mechanism proposed in [9] is a combination of the above two methods. It considered the population's distributions in the fitness space and search space simultaneously and hereafter derived the third group of adaptation strategy.

This paper focuses on the idea of the third group. In addition, different from [9] which consider the best and the worst individuals in each generation to estimate evolution stage and adapt the parameter, this paper proposes a novel strategy concerning the set of individuals which have good fitness values and close to the found best solution. This strategy can achieve a more comprehensive estimation of evolution stage by considering a set of good individuals. Then a set of rules are designed to adapt the parameter and operation simultaneously according to the evolution state.

As we know, individuals in the population will crowd together during the evolution. In a smooth fitness landscape the individuals which are close to the best individual will have considerably better fitness values during the evolving and converging process of EAs. Based on the above phenomenon,

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

the proposed method uses individuals which are close to the best-so-far solution to compose a location neighboring set (LNS) in each generation of EAs. It can be noted that LNS reflects the density of the population distribution. On the other hand, the proposed method selects the best individuals in the current population to build a fitness neighboring set (FNS) that has identical size with the LNS. In this way, FNS reflects the mature degree of the population distribution in the fitness space. The joining of the two sets are the individuals those not only are close to the best individual but also have good fitness values. The size of the joining set will become larger during the evolving process. Therefore, we can define the maturity degree of the population by considering the size of the joining set. Then the evolution state is determined via a set of thresholds according to the obtained maturity degree. Accordingly, the proposed maturity-based adaptation strategy uses an adaptive variation to adjust the parameter values based on the adaptation rules designed for the present evolution state. Meanwhile, a novel allele gene-based mutation is developed in this paper. Based on our proposed adaptation rules, this mutation is executed in some certain evolution state instead of the traditional uniform mutation in order to accelerate the convergence speed. Therefore, in this paper, not only the parameter settings but also the genetic operator is adaptively adjusted according to the maturity degree of the population.

In the experiment, the proposed maturity-based adaptation strategy is implemented on GA for adjusting p_x, p_m and the mutation operator. A maturity-based adaptive GA that applies the proposed adaption strategy to the standard GA (SGA) is validated over a set of eight benchmark functions with different characteristics. By comparing the experimental results of maturity-based adaptive GA with those of SGA, the effectiveness and efficiency of the proposed adaption strategy can be verified.

The rest of this paper is organized as follows. Section II gives a brief review on EAs. Section III presents the proposed method for determining the maturity degree and the proposed adaptation strategy of p_x, p_m and the mutation operator. Experimental results on eight benchmark functions are displayed in Section IV. Section V draws a conclusion to the whole paper.

II. Brief Review on EAs

EAs are generic population-based metaheuristic optimization algorithms. Although there are many different variants of EAs, the underlying idea is the same. EAs generate a population of individuals (represent candidate solutions to the optimization problem) and select them according to fitness function. The fitness function determines the environmental pressure of each individual. Evolution of the population then takes place and obtains more favorable fitness values after the repeated application of reproduction (like mutation and recombination) and selection. GAs are one of the most famous algorithms of EAs. In this paper, we take GAs as an example and discuss GAs in details.

GAs are population-based probabilistic optimization techniques that simulate the evolutionary process of biological organisms. In GAs, each individual represents a possible solution to the optimization problem. The fitness of the individual, measured by a fitness function, judges the quality of the solution that the individual stands for. In each generation, the population is evolved through a series of evolution operators that mimic the reproduction and selection mechanisms in nature. Based on the principle of 'survival of the fittest' in evolutionary theory, individuals with better fitness have a larger chance to survive and reproduce. Suppose that the optimization problem has D variables. The generic procedure of GAs can be summarized as follows.

Step 1) Initialization:

In this step, the control parameters of GAs, including the population size N, the crossover probability p_x, and the mutation probability p_m, are first determined. Then an initial population is formed with each individual $\boldsymbol{x}_i$ generated as a random solution in the search space, i=1,2,…,N. The N individuals are evaluated by the fitness function f and the best-so-far solution is denoted as f_{best}.

Step 2) Selection:

The selection step aims to give more chances to individuals with better fitness to survive and reproduce. By doing so, the good components (gene segment) in good individuals can be inherited and further developed. A number of selection methods such as remainder stochastic sampling [10], proportional selection [11], linear rank selection [11] and uniform ranking [12] have been proposed in the literature. In this paper, the canonical tournament selection is adopted. In detail, the tournament selection runs N times of "tournaments". In each tournament, a subset of M individuals is randomly selected from the population and the individual with the best fitness is regarded as the winner. The N winners compose the mating population.

Step 3) Crossover:

Individuals in the mating population exchange their genes with each other to produce offspring. By recombining good individuals that survive after the selection step, better individuals are likely to be generated. Crossover methods with different characteristics have been proposed for different problems, including one-point crossover [13], two-point crossover [14], multiple-point crossover [14], [15] and shuffle crossover [14]. This paper employs the most commonly used method, one-point crossover. In this method, two individuals are randomly selected from the mating population with the crossover probability p_x. Two offspring are created by exchanging the genes of the two selected individuals after a randomly selected locus. The procedure repeats until N new individuals are created.

Step 4) Mutation:

The purpose of mutation is to maintain the population diversity and reduce the chance of premature convergence. The mutation can also generate the better individuals in some

cases. When mutating an individual $\boldsymbol{x}_i$, a random number p_j ($p_j \in [0,1]$) is first generated for each gene x_i^j in $\boldsymbol{x}_i$, j=1,2,…,D. If p_j is smaller than the mutation probability p_m, the gene x_i^j undergoes mutation and is changed to a random value in the feasible domain of x_i^j.

Step 5) Elitist Strategy:

Calculate the fitness of each chromosome and identify the best individual C_B and the worst individual C_W in this generation. If C_B is better than the historic best individual, f_{best}, update f_{best} with C_B. Otherwise, replace C_W with f_{best}.

Step 6) Termination Check:

If the termination criterion, says, the maximum number of generations, has been reached, the algorithm terminates and returns the best-so-far solution as the result. Otherwise, the algorithm returns to Step 2) to start a new generation.

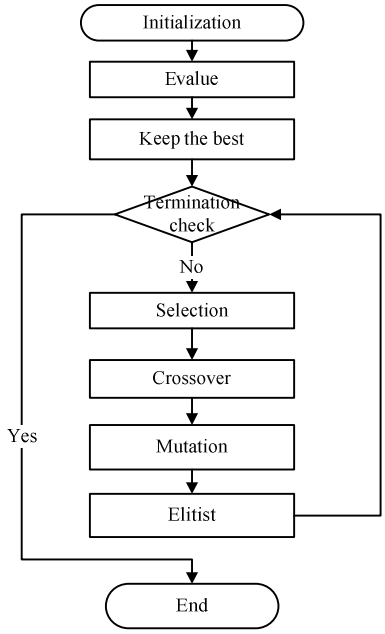


Figure 1. Flowchart of the generic procedure of GAs.

A flowchart of the above general procedure of GAs is depicted in Fig. 1. From the above, it can be observed that the steps of selection, crossover and mutation play important roles in GAs. So the parameters and the method of the crossover and mutation must be carefully selected. The appropriate values and operator can help to accelerate the converge speed and find a better solution for the specific question.

III. Adaptive Strategy

In canonical GAs, p_x and p_m are usually set to fixed values by empirical experiences and the method of mutation are often remain the same in different states. However, it is more desirable that the value of p_x and p_m can be adapted to the evolution states and different operator can be employed to meet the needs in different phase of the optimization procedure. In this paper, four evolution states are defined, i.e., un-mature state, sub-maturing state, maturing state and matured state. In each generation, the present evolution state is first determined based on the population distribution in both the search space and the fitness space. Then the values of p_x and p_m are adjusted according to the adaptation rules. The allele gene-based mutation and traditional mutation are executed alternately in different evolution states. Detailed implementation of the state identification method and the design of adaptation rules are introduced as follows.

A. Identification of Evolution States

This paper proposes to identify the evolution states by analyzing the distribution of good individuals. The optimization procedure is considered to have a higher maturity degree when more and more good individuals are gathering around the best-so-far individual $\boldsymbol{x}_b$. In detail, the state identification method comprises the following steps.

Step 1) Calculate the mean distance of the other individuals in the population to the best-so-far individuals, i.e.,

$$d_{mean} = \frac{1}{N-1}\sum_{i=1}^{N}\sqrt{\sum_{k=1}^{D}\left(x_i^k - x_b^k\right)^2}\ , \qquad (1)$$

x_i^k and x_b^k stand for the value on the k-th dimension of the i-th individual $\boldsymbol{x}_i$ and the best individual $\boldsymbol{x}_b$, k=1,2,…,D, i=1,2,…,N and N denotes the number of individuals in the population.

Step 2) Identify the set $\boldsymbol{A}$ of individuals whose distances to $\boldsymbol{x}_b$ is smaller than d_{mean}. Mathematically, the set $\boldsymbol{A}$ can be written as

$$A = \{\mathbf{x}_i \mid d_i \le d_{\text{mean}}, i = 1,2,...,N\}, \qquad (2)$$

where d_i denotes the Euclidean distance from $\boldsymbol{x}_i$ to $\boldsymbol{x}_b$. As exemplified in Fig. 2, an individual in $\boldsymbol{A}$ can be considered to be relatively close to $\boldsymbol{x}_b$ in the search space if its distance to $\boldsymbol{x}_b$ is smaller than d_{mean}. Denote the number of individuals in $\boldsymbol{A}$ with $|\boldsymbol{A}|$. For example, in Fig. 2 the individuals inside the circle belong to $\boldsymbol{A}$.

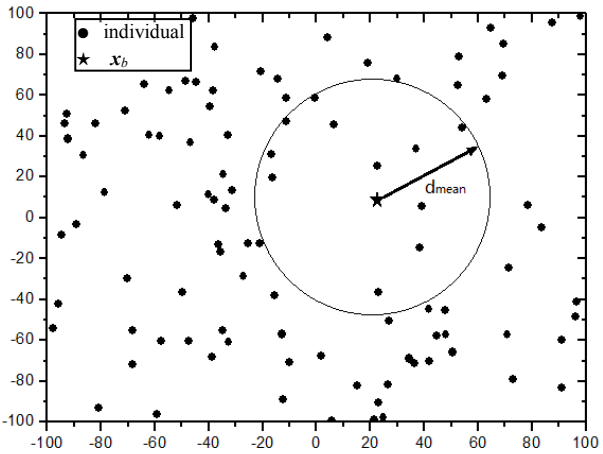


Figure 2. An example of the set A of individuals close to the best-so-far solution $\boldsymbol{x}_b$.

Step 3) Identify the set $\boldsymbol{G}$ of individuals that are close to $\boldsymbol{x}_b$ in the fitness space. For doing this, the individuals in the

population are sorted in order of ascending fitness values (in case of minimization problems). The first $|\boldsymbol{A}|$ individuals compose the set $\boldsymbol{G}$. As the fitness values have been calculated and stored in the vector before, the sorting process doesn't have any evaluation times.

Step 4) Identify the set $\boldsymbol{C}$ of individuals that are close to $\boldsymbol{x}_b$ in both the search space and the fitness space as

$$C = A \cap G \tag{3}$$

Step 5) The maturity degree μ of the optimization procedure is then defined as

$$\mu = (1 - P_r) \times \frac{|C|}{N}, \tag{4}$$

where $|\boldsymbol{C}|$ denotes the number of individuals in $\boldsymbol{C}$, and therefore $|\mathbf{C}|/N$ stands for the proportion of the set $\boldsymbol{C}$ in the population. P_r, which represents the percentage of search space that the LNS occupied, is defined as

$$P_r = \frac{d_{mean}}{r_s} \tag{5}$$

Here r_s is the radius of the search space, d_{mean} is the mean distance of the other individuals in the population to the best-so-far individuals.

$$r_s = \sqrt{\sum_{k=1}^{D} \left(s_{\max}^k - s_{\min}^k\right)^2} \tag{6}$$

In (6), $s_{\min}^k$ is the lower bound of the search space and $s_{\max}^k$ is the upper bound of the search space in k-th dimension.

From the above, it can be observed that the value of μ is in the range of [0, 1] and it increases as the set $\boldsymbol{C}$ enlarges. The optimization procedure thus gains a higher maturity degree when more individuals are getting close to $\boldsymbol{x}_b$ (the best so far individual).

Step 6) Determine the evolution state according to the μ value. The feasible range of the μ value is classified into four categories S1, S2, S3, and S4, which correspond to the optimization states of un-mature, sub-mature, maturing, and matured, respectively. The set of thresholds are used to determine the category the individuals belong to.

When initialize the population, genes are randomly generated. Therefore, the number of genes which are within the circle is approximate to half of the population size. In this situation, μ is around to 0.5 so $\mu \in [0,0.5]$ is defined as S1. The other state are define as following: $\mu \in (0.5,0.666]$ is defined as S2, $\mu \in (0.666,0.832]$ is S3 and $\mu \in (0.832,1]$ is S4. The interval value of μ is the same and equals to 1.666 = (1-0.5)/3.

B. Maturity-based Adaptation Strategies of p_x and p_m

In different states, p_{x} and p_{m} are adjusted according to different rules. In the states of un-mature and sub-mature, the main purpose is to accelerate search speed. In the states of maturing and matured, the main purpose becomes avoiding premature convergence. In order to achieve both goals of accelerating the convergence speed and increasing the chance to jump out of local optima, we design the maturity-based adaptation rules of p_{x} and p_{m} as follow.

TABLE I. ADAPTATION RULES OF PX AND PM IN DIFFERENT OPTIMIZATION STATES

Un-mature	**Sub-mature**	**Maturing**	**Matured**
p_{m} decrease	p_{x} increase	p_{x} decrease	p_{m} increase

Rule 1 Decrease p_{m} by ∇p_{m} in the un-mature state.

There are two alternatives for accelerating the convergence speed: decreasing p_{m} and increasing p_{x}. However, the consequences generated by these two alternatives are different. Increasing p_{x} gives more chances for individuals to crossover. Crossover between two good individuals can create a promising individual and thus accelerate the convergence to the global optimum. However, crossover between two unsatisfying individuals may not bring such a satisfying result. Since individuals are not supposed to have good solution quality at the beginning of the algorithm, increasing p_{x} in un-matured state will not get many good results. In contrast, decreasing p_{m} gives individual less chance to mutate and the algorithm is easier to converge. Since the individuals are already diverse in the un-mature state, the value of p_{m} can decrease to help accelerate the convergence speed.

Rule 2 Increase p_{x} by Δp_{x} in the sub-mature state.

In the sub-mature state, the population starts to converge towards the best-so-far solution. Increasing p_{x} can speed up such convergence by allowing more individuals carrying good genes to crossover. However, since the best-so-far solution may be a local optimum, decreasing p_{m} in the sub-mature state will do harm to the population diversity and hence cause early convergence. Consequently, increasing p_{x} in the sub-mature state is a better choice for accelerating the convergence speed in the sub-mature state.

Rule 3 Decrease p_{x} by ∇p_{x} in the maturing state.

There are also two alternatives to avoid prematurity: decreasing p_{x} and increasing p_{m}. In the maturing state, the individuals are swarming around the best-so-far individual and getting similar. If p_{x} is too high, the population will quickly lose the diversity and the algorithm will be easily trapped in local optima. It is therefore necessary to decrease the p_{x} in the maturing state.

Rule 4 Increase p_{m} by Δp_{m} in the matured state.

In the matured state, most of the population has converged at the best-so-far individual. If the best-so-far individual locates on a local optimum, increasing p_{x} can hardly help the algorithm to escape from the local optimum due to the low diversity of the population. In contrast, increasing p_{m} can increase the population diversity effectively and thus give

more opportunities for the population to jump out of the local optimum and search for the global optimum.

After defining the above four rules, a remaining problem in the proposed maturity-based adaptation strategies is how to define the increments and decrements of p_x and p_m. To better suit the optimization procedure, it is more desirable that the speed of adjusting p_x and p_m is also adapted to different optimization states. Specifically, the settings of Δp_x, Δp_m, ∇p_x, and ∇p_m are showed as below.

To accelerate the convergence speed in the un-mature and sub-mature states, the settings of Δp_x and ∇p_m are formulated as follows:

$$\Delta p_x = K_x e^{1-\mu} \tag{7}$$

$$\nabla p_m = K_m e^{1-\mu}, \tag{8}$$

where K_x and K_m are both chosen to keep the changes of the values of p_x and p_m within a tolerance percentage of the nominal level in each generation. K_x corresponds to the precision of p_x and K_m corresponds to the precision of p_m. When the population is scattered in the un-matured state, the values of p_m should be adapted greatly in order to immediately respond to their required values for current circumstances, and thus to increase the speed of convergence. When population is sub-maturing, the values of p_x should be modified slightly to sustain in a suitable range, so that they can continue the trend of convergence.

To avoid the local optimality in the last two stages, the settings are defined according to the following two formulas:

$$\nabla p_x = K_x e^{2\mu} \tag{9}$$

$$\Delta p_m = K_m e^{2\mu} \tag{10}$$

When the population is in maturing or matured state, the values of p_x and p_m should be modified according to the maturity degree. The value of p_x should decrease faster to protect the population from prematurity when the population is more mature. And the value of p_m should increase to the required values more rapidly to jump out of the local optimum when the population is denser.

It should be noted that p_x and p_m have limits. For example, $p_x \in [0,1]$, $p_m \in [0,0.1]$.

C. Maturity-based Adaptation Strategies of Mutation

In this paper, not only the p_x and p_m are adapted according to the evolution stage, but also the mutation operator of GA. We use the traditional random mutation in S1 (to do exploration) and S4 (to jump out from local optimum), and adopt a novel allele gene-based mutation in S2 and S3 (to avoid breaking the already built population structure so as to accelerate the convergence). The allele gene-based mutation scheme is introduced and described as follows.

In each generation, the alleles in each dimension j are regarded as a vector y_j**.** Vector y_j has N dimensions and N is the size of the population. In this way, the search space of GA can be regarded as a vector space $\boldsymbol{Y} = (y_1, y_2, \ldots, y_D)$ where D is the number of variables. The mutation selects several pairs of alleles in an allele vector y_j each time. And the mutation performs on each pair of alleles. The procedures of the mutation are described as follows.

Step 1) Alleles in the same pair are selected from the same vector y_j (j=1, 2, …, D). In order to determine how alleles form a pair, a random number r_{ij} ($r_{ij} \in [0,1]$) is generated for each allele y_j^i(j=1, 2, …, D, i=1, 2, …, N,). If r_{ij} is smaller than the mutation probability p_m, the allele will be selected as the first element of the pair. If the next allele y_j^{i+k} ($k \in \mathbf{Z}$, $k \in [1, N-i]$) has a random number $r_{(i+k)j}$ smaller than p_m, it will be selected as the second element of the pair. The process continues until all the alleles are considered. In this step, pairs of alleles are formed to undergo mutation.

Step 2) For each pair, suppose two genes y_j^i and y_j^{i+k} are selected to undergo mutation. Define R as a bivector and $R = (y_j^i - c, y_j^{i+k} - c)$, where c is the center of the search space for j-th variable, i.e., $c = \frac{1}{2}(l, u)$, l represents the lower bound of search space and u represents the upper bound. Then, R is multiplied by a matrix M, i.e., $R' = (y_j^{i\prime}, y_j^{i+k\prime}) = MR$. After that, the selected genes y_j^i and y_j^{i+k} are replaced by $y_j^{i\prime}+c$ and $y_j^{i+k\prime}+c$ respectively. In order to treat every individual without bias, M should be an orthogonal matrix, i.e., $MM^T = I$. In our experiment, we use a randomly generated 2D rotation matrix as

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{11}$$

where θ is a random value uniformly distributed over $[-\pi, \pi]$.

The adaptive strategy proposed above can improve the behavior of GA significantly. Experiments and discussions of GA with the adaptive strategy are presented in the next section.

IV. Experiments and Discussions

In this section, we implement a maturity-based adaptive GA (MBAGA) by incorporating the proposed maturity-based adaptation strategy of p_x and p_m into each generation of the standard GA (SGA). The MBAGA is applied to a set of eight benchmark functions [16]. The experimental results are compared with those of SGA for illustrating the effectiveness and efficiency of the proposed maturity-based adaptation strategy.

A. Experimental Settings

Table II summarizes the benchmark functions used in experiments. Among the eight functions, f_1 to f_4 are unimodal functions. Function f_4 is the step function, which has one minimum and is discontinuous. Functions f_5 to f_8 are multimodal functions where the number of local minima increases exponentially with the problem dimension. All the functions have 30 dimensions. And except for f_5 has global minimum value at -12569.5, other functions have minimum values at zero.

For fair comparison, the common parameters of MBAGA and SGA are set the same. Specifically, the population size and the tournament size of MBAGA and SGA are set as 100 and 10, respectively. Both algorithms terminate after 3000

generations and are run for 30 independent times. Besides, the initial values of p_x and p_m in MBAGA are set the same as the fixed values of p_x and p_m in SGA.

TABLE II. BENCHMARK FUNCTIONS

Function	Search Domain
$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100,100]^{30}$
$f_2(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10,10]^{30}$
$f_3(\mathbf{x}) = \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$	$[-100,100]^{30}$
$f_4(x) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2$	$[-100,100]^{30}$
$f_5(\mathbf{x}) = -\sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500,500]^{30}$
$f_6(\boldsymbol{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]^{30}$
$f_7(\mathbf{x}) = -20\exp\left[-0.2\sqrt{\sum_{i=1}^{D} x_i^2 / D}\right] - \exp\left[\sum_{i=1}^{D} \cos(2\pi x_i) / D\right] + 20 + e$	$[-32,32]^{30}$
$f_8(\mathbf{x}) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, $y_i = 1 + (x_i + 1)/4$	$[-50,50]^{30}$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

B. Comparison with SGA

Table III and Table IV tabulates the mean values over 30 trials obtained by MBAGA and SGA (with elitism) under different combinations of p_x and p_m. As shown in the table, the value of p_x is chosen from the set of {0.3, 0.6, 0.9}, while the value of p_m is chosen from {0.03, 0.06, 0.09}. Consequently, there are totally (3×3=9) combinations of p_x and p_m. Once a combination is chosen, the values of p_x and p_m remains unchanged during the optimization procedure of SGA, while the MBAGA uses the combination as the initial settings and adapts the values of p_x and p_m according to the evolution state.

The comparison in Table III and Table IV shows that for all of the unimodal and multimodal functions, MBAGA achieves much more favorable results than SGA does. Specifically, when the p_x and p_m don't suit for the problem. For example, in f_1 the value obtained by MBAGA is 10^{85} times smaller than SGA. The advantages of MBAGA confirm that the proposed maturity-based adaptation strategy of p_x and p_m can improve the performance of SGA in terms of solution accuracy.

TABLE III. COMPARISON BETWEEN SGA AND MBAGA ON UNIMODAL

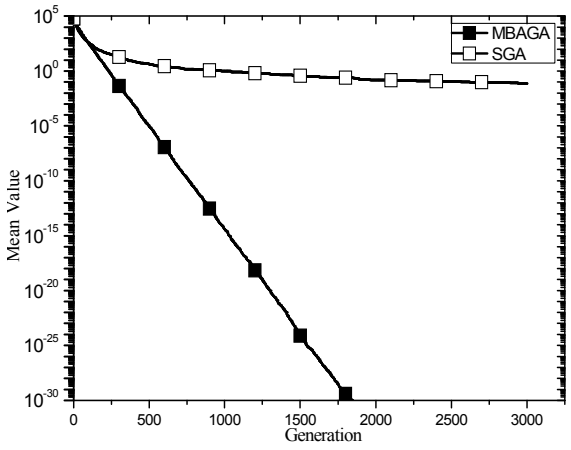
(Px,pm)	GAS	f_1	f_2	f_3	f_4
(0.3,0.03)	SGA	0.054292	0.0847804	1.39714	0
	MBAGA	**3.46938e-087**	**4.44092e-056**	**1.05095e-014**	**0**
(0.3,0.06)	SGA	0.16494	0.154485	1.71392	0
	MBAGA	**1.72363e-087**	**1.6474e-056**	**1.74802e-014**	**0**
(0.3,0.09)	SGA	5.34477	0.829981	4.95621	6.8
	MBAGA	**5.67549e-087**	**9.10113e-057**	**9.30342e-015**	**0**
(0.6,0.03)	SGA	0.046826	0.0740379	1.20487	0
	MBAGA	**1.03813e-087**	**1.43914e-056**	**4.6321e-015**	**0**
(0.6,0.06)	SGA	0.183202	0.140507	1.71026	0
	MBAGA	**1.82952e-087**	**2.23042e-056**	**1.11146e-014**	**0**
(0.6,0.09)	SGA	6.04869	0.713279	5.26139	7
	MBAGA	**1.46367e-087**	**1.83541e-056**	**5.72403e-015**	**0**
(0.9,0.03)	SGA	0.0416401	0.0836818	1.19083	0
	MBAGA	**2.4644e-087**	**1.20962e-056**	**2.2908e-014**	**0**
(0.9,0.06)	SGA	0.162481	0.13184	2.15217	0
	MBAGA	**8.778e-088**	**2.35437e-056**	**5.0474e-015**	**0**
(0.9,0.09)	SGA	6.65517	0.611347	5.06607	6.3
	MBAGA	**7.58062e-088**	**6.21936e-057**	**4.0122e-014**	**0**

TABLE IV. COMPARISON BETWEEN SGA AND MBAGA ON MULTIMODAO

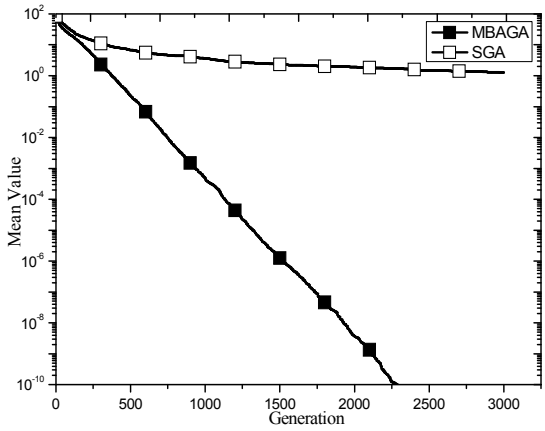
(Px,pm)	GAS	f_5	f_6	f_7	f_8
(0.3,0.03)	SGA	-12569.3	0.0261572	0.0561668	0.000206161
	MBAGA	**-12569.4**	**0**	**4.35207e-015**	**1.93807e-007**
(0.3,0.06)	SGA	-12568.9	0.0869569	0.125679	0.00107914
	MBAGA	**-12569.3**	**0**	**4.70735e-015**	**2.32791e-007**
(0.3,0.09)	SGA	-12558.8	1.80336	1.28174	0.0221142
	MBAGA	**-12569.3**	**0**	**4.70735e-015**	**7.65771e-007**
(0.6,0.03)	SGA	**-12569.3**	0.026843	0.0642519	0.000260119
	MBAGA	**-12569.3**	**0**	**4.70735e-015**	**4.47139e-007**
(0.6,0.06)	SGA	-12569.1	0.063947	0.122183	0.000614699
	MBAGA	**-12569.3**	**0**	**4.70735e-015**	**4.36212e-007**
(0.6,0.09)	SGA	-12560.5	1.36908	1.04897	0.0357618
	MBAGA	**-12569.3**	**0**	**4.35207e-015**	**2.99656e-007**
(0.9,0.03)	SGA	**-12569.3**	0.0201669	0.0619914	0.000163244
	MBAGA	**-12569.3**	**0**	**4.35207e-015**	**7.5529e-007**
(0.9,0.06)	SGA	-12569.1	0.0604329	0.101188	0.000613264
	MBAGA	**-12569.3**	**0**	**5.06262e-015**	**7.62812e-007**
(0.9,0.09)	SGA	-12563.5	1.25782	1.23098	0.0303506
	MBAGA	**-12569.4**	**0**	**3.9968e-015**	**4.22825e-007**

Meanwhile, it can be observed that the MBAGA achieves similar results even if p_x and p_m are initialized with different values. For example, the results for SGA in f_7 can achieve 0.0561668 when the initial p_x and p_m are desirable. But the results go bad to 1.23098 when the initial p_x and p_m don't suit for the problem. However, the results of MBAGA are similar and in the same order of magnitude in different initial settings of p_x and p_m. Such a phenomenon shows that MBAGA is insensitive to the initial settings of p_x and p_m. Therefore, it can be concluded that the proposed maturity-based adaptation strategy increases the robustness of SGA.

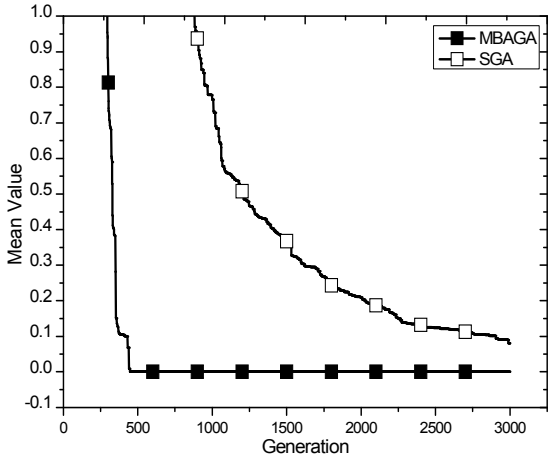
The convergence curves of mean values during the training process in solving the f_1, f_3, f_6 and f_8 are shown in Fig. 3, where the vertical axis is the mean function value of 30 trials and the horizontal axis is the number of generation.



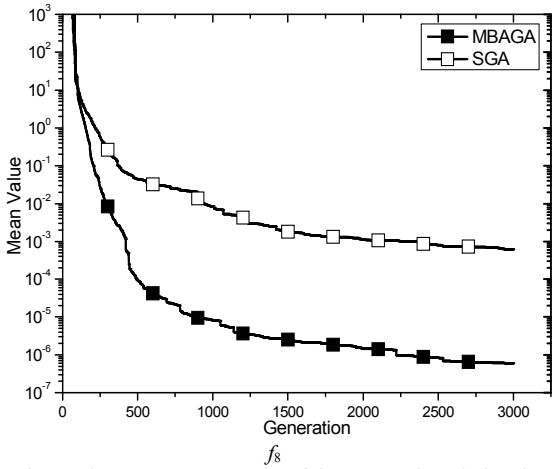
f_1



f_3



f_6



f_8

Fig. 3. The convergence curves of the mean values during the training process

As Fig. 3 shows, MBAGA displays a faster convergence speed than SGA. The acceleration of convergence is due to the adaptive values of p_x and p_m and the suitable mutation method in different evolution states. It is obvious that the MBAGA achieves more favorable results and maintains a fast convergence speed all along the optimization process in all of the four functions.

The results in Table V also show that the proposed algorithm out runs many other adaptive algorithms in EA. Table V tabulates the mean values over 30 trials obtained by MBAGA, APSO[17] and SaDE[18]. It shows that MBAGA performs well especially in the multimodal functions. For unimodal function, MBAGA outperforms others except that APSO is better on f_1 and f_2 (the data for APSO and SaDE is obtained in [17] and [19]).

TABLE V. COMPARISON BETWEEN MBAGA, APSO AND SADE

function	**MBAGA**	**APSO**	**SaDE**
f_1	7.58062e-088	**1.45e-150**	3.02e-23
f_2	6.21936e-057	**5.73e-83**	4.64e-11
f_3	**5.0474e-015**	3.9e-011	1.59e-03
f_4	**0**	**0**	**0**
f_5	**-12569.5**	**-12569.5**	**-12569.5**
f_6	**0**	-5.8e-15	4.0e-8
f_7	**3.9968e-015**	1.11e-14	9.06e-11
f_8	**1.93807e-007**	3.76e-31	1.21e-19

V. CONCLUTION

This paper proposes a novel maturity-based adaptation strategy to adjust the crossover and mutation probabilities as well as the operator of EAs. Implemented on GA, the proposed maturity-based adaptation strategy determines the maturity degree of the optimization procedure by analyzing the distribution of good individuals in the population. The more the good individuals concentrate around the best-so-far solution, the higher the maturity degree becomes. According to the maturity degree, adaptation rules are chosen to adjust the values of p_x and p_m for satisfying the desire for crossover and mutation at the present optimization state. Meanwhile a novel allele gene-based mutation is executed to accelerate the convergence speed in certain evolution states according to the adaptive rules. Experiments on eight benchmark functions show that the proposed maturity-based adaptation strategy can improve the solution accuracy, search speed, and robustness of GAs.

VI. FUTURE WORK

As future trends for further exploration of the adaptive strategy proposed in this paper, it can be applied to Particle Swarm Optimization (PSO) [20][21], Ant Colony Optimization (ACO) [22][23] and Differential Evolution (DE)[24] to adapt the parameters and operator based on maturity degree.

REFERENCES

[1] Carlos A. Coello Coello, David A. Van Veldhuizen, Gary B. Lamont, *Evolutionary algorithms for solving multi-objective problems,* Springer, 2002.

[2] Bäck, T., *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University, 1996.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 1992.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.

[5] Jun ZHANG, Zhi-hui Zhan, Ying Lin, Ni Chen, Yue-jiao Gong, Henry S.H. Chung, Yun Li and Yu-hui Shi, "Evolutionary Computation Meets Machine Learning: A Survey", *IEEE Computational Intelligence Magazine*, pp.68-75, NOVEMBER 2011

[6] M.Srinivas, and L.M.Patnaik, "Adaptive probabilities of crossover genetic in mutation and algorithms," *IEEE Transactions on systems , man and cybernetics*, vol. 24, April 1994, pp. 656-667

[7] C. H. Dai ,Y. F. Zhu, and W. R. Chen, "Adaptive probabilities of crossover and mutation in genetic algorithms based on cloud model," *Proceedings of 2006 IEEE Information Theory Workshop (ITW'06)*, pp. 710-713

[8] Xiaoming Dai, "Allele gene based adaptive genetic algorithm to the code design," *IEEE Transactions on communications*, vol. 59, No. 5, May 2011, pp. 1253-1258

[9] Jun ZHANG, Henry Chung and W.L., LO, "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms", *IEEE Transactions on Evolutionary Computation* Vol.11, No.3, June 2007, Page. 326-335

[10] James E. Baker, "Reducing bias and inefficiency in the selection algorithm," *In Proceedings of the Second International Conference on Genetic algorithms and their application*, John J. Grefenstette (Ed.). L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 14-21. 1987.

[11] J. J. Grefenstette, and J. E. Baker, "How genetic algorithms work: a critical look at implicit parallelism," *in Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 20-27.

[12] T. Back, and F. Hoffmeister, "Extended selection mechanisms in genetic algorithms," *in Proc. 4th Int. Conf. Genetic Algorithms*, Univ. of California, San Diego, CA, 1991, pp. 99-99.

[13] J. H. Holland, *Adaptation in natural and artificial systems*, Ann Arbor, MI: University of Michigan Press, 1975. viii, 183 pp.

[14] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, "Biases in the crossover landscape," *in Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 10-19.

[15] W. M. Spears, and K. A. DeJong, "An analysis of multi-point crossover,'' *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. pp. 301-315, 1991.

[16] Xin Yao, Yong Liu, and Guangming Lin, "Evolutionary programming made faster," *IEEE Transaction on evolutionary computation*, vol. 3, No. 2, July 1999, PP: 82-102

[17] Zhi-Hui Zhan, Jun Zhang, and Yun Li, "Adaptive Particle Swarm Optimization" *IEEE Transaction on systems, man and cybernetics --Part B,* vol. 39, No. 6, December 2009, pp:1362-1381

[18] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791, 2005.

[19] Zhenyu Yang, Ke Tang and Xin Yao, "Self-adaptive Differential Evolution with Neighborhood Search" *In Proceedings of 2008 IEEE Congress on Evolutionary Computation,*pp:1110-1116

[20] J.Kennedy, R.Eberhart, "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp.1942–1948.

[21] W.N. Chen, Jun ZHANG, Henry Chung, W.L., Zhong, W.G. Wu and Y.H., Shi, "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems", *IEEE Transactions on Evolutionary Computation,* vol.14,No.2, April 2010, pp.278-300

[22] M. Dorigo, V. Maniezzo, et A. Colorni, "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics--Part B* , vol. 26, No. 1, 1996, pp: 29-41.

[23] W.N. Chen and Jun ZHANG, "Ant Colony Optimization Approach to Grid Workflow Scheduling Problem with Various QoS Requirements", *IEEE Transactions on Systems, Man, and Cybernetics--Part C: Applications and Reviews,* vol. 31, No. 1, Jan 2009, pp.29-43

[24] Rocca, P.,Oliveri, G., Massa, A. "Differential Evolution as Applied to Electromagnetics". *IEEE Antennas and Propagation Magazine* vol 53, Issue.1, 2011, pp: 38–49.