

# Immigrant schemes for evolutionary algorithms in dynamic environments: Adapting the replacement rate

YU Xin<sup>1</sup>, TANG Ke<sup>1\*</sup> & YAO Xin<sup>1,2</sup>

<sup>1</sup>*Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China,*

<sup>2</sup>*Centre of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K.*

Received January 16, 2009; accepted April 1, 2010

---

**Abstract** One approach for evolutionary algorithms (EAs) to address dynamic optimization problems (DOPs) is to maintain diversity of the population via introducing immigrants. So far all immigrant schemes developed for EAs have used fixed replacement rates. This paper examines the impact of the replacement rate on the performance of EAs with immigrant schemes in dynamic environments, and proposes a self-adaptive mechanism for EAs with immigrant schemes to address DOPs. Our experimental study showed that the new approach could avoid the tedious work of fine-tuning the parameter and outperformed other immigrant schemes using a fixed replacement rate with traditionally suggested values in most cases.

**Keywords** evolutionary algorithm, dynamic optimization problem, immigrant scheme, self-adaptive replacement rate

---

**Citation** Yu X, Tang K, Yao X. Immigrant schemes for evolutionary algorithms in dynamic environments: Adapting the replacement rate. *Sci China Inf Sci*, 2010, 53: 1–11, doi:

---

## 1 Introduction

There are many challenging dynamic optimization problems (DOPs) in real-world applications. One of the best known examples is the fluctuations of the stock price over time during the process of maximizing the expected yields. Evolutionary algorithms (EAs) are good candidates for addressing DOPs since they mimic the process of natural evolution which copes very well in changing environments. However, traditional EAs tend to be easily trapped in a static environment, since the population will converge to a point of the search space, and once the environment changes, it is hard for the individuals to jump to the new promising regions of the search space. Commonly used methods to enhance the EAs' performance in dynamic environments include re-introducing and maintaining the diversity, memory schemes and multi-population schemes [1]. While a substantial emphasis has been put on the continuous domain, Rohlfshagen and Yao [2] recently analyzed dynamics in the combinatorial domain via an example of the subset sum problem. They showed that some assumptions traditionally regarded as being reasonable for developing new algorithms did not necessarily hold in the case of discrete optimization.

---

\*Corresponding author (email: ketang@ustc.edu.cn)

Among approaches for DOPs, immigrant schemes have proved to be effective via maintaining the diversity of the population throughout the run [3 – 7]. They work by replacing a predefined proportion of the population with the newly generated immigrants at the end of each generation. When designing an immigrant scheme, generally there are four issues needing to be addressed, namely the mechanism of generating immigrants, the replacement strategy, the survival capability of the introduced immigrants in the population, and the replacement rate [3]. Over the years, researchers have already done some research on the first three issues, e.g., refs. [4, 5, 6] for the mechanism of generating immigrants, ref. [7] for the replacement strategy and the survival capability of the new immigrants in the population, whereas there has been little work on the replacement rate.

The replacement rate represents the proportion of the population that will be replaced by immigrants. All previously designed immigrant schemes simply set the replacement rate to a fixed value, e.g., 0.2 [4, 5] or 0.3 [6]. In this paper, the impact of the replacement rate on the performance of EAs with immigrant schemes in dynamic environments is examined, and a self-adaptive scheme is designed for immigrant schemes. It is known that the self-adaptation has been widely and successfully applied to the field of evolutionary computation (EC) [8], such as genetic algorithms (GAs) [9, 10, 11], evolutionary programming (EP) [12], evolution strategy (ES) [13], differential evolution (DE) [14], and particle swarm optimization (PSO) [15]. However, these attempts all aimed at optimization problems in stationary environments.

The main contributions of this paper are listed as follows. First, we fill the gap of research on the replacement rate for immigrant schemes. Second, we eliminate the parameter replacement rate. Although we introduce a new parameter, experimental results showed that the performance of the algorithm was much less sensitive to the new parameter than to the replacement rate. Third, the performance when using the self-adaptive mechanism outperformed that when traditionally suggested values (i.e., 0.2 and 0.3) of the replacement rate were used in most cases. Furthermore, the self-adaptation avoids fine-tuning the parameter replacement rate to achieve the best performance in different kinds of environment. There is only a slight degradation in the performance.

The rest of this paper is organized as follows: Section 2 briefly reviews immigrant schemes for EAs in dynamic environments. Section 3 presents the self-adaptive replacement rate for immigrant scheme. Section 4 describes the experimental setup for this study. The experimental results and analysis are presented in Section 5. Finally, conclusions are drawn in Section 6.

## 2 Immigrant schemes for DOPs

Classic EAs tend to converge to a single solution, and when the environment changes, it takes some time for the population to adapt to the new environment and move towards the new promising region, because nearly all the individuals of the population concentrate on a point of the search space. To address this problem, immigrant schemes are introduced into EAs to maintain the diversity of the population throughout the run [3 – 7, 16 – 18]. They work by replacing a predefined proportion of the population with newly generated immigrants at the end of each generation.

The most prominent approach to generate immigrants seems to create immigrants randomly. Grefenstette [16] used randomly generated individuals to replace the worst individuals of the population in each generation. The random immigrant scheme works on the analogy of the flux of immigrants that wander in and out of a population between two generations in nature [17]. His study shows that the random immigrant scheme works well in environments where there are occasional, large changes in the location of the optimum.

Branke [19] stated that continuous adaptation only made sense when problems to be studied featured “small to medium” environmental changes, otherwise to restart the search from scratch would be the proper choice. Under this assumption, compared with randomly generating immigrants, it would be more suitable to generate immigrants based on the information of the population, since individuals in the previous environment may still be quite fit in the new environment.

According to the way in which the information of the population is used, existing immigrant schemes

are categorized into the *direct immigrant scheme* and the *indirect immigrant scheme*, respectively [3]. The *direct immigrant scheme* generates immigrants based on the current population. On the other hand, the *indirect immigrant scheme* first builds a model based on the current population, then generates immigrants according to the model. Examples of the *direct immigrant scheme* include the elitism-based immigrant scheme [5] in which immigrants come from mutating the elite from previous generation, and a hybrid immigrant scheme combining the elitism-based, the traditional random, and the dualism-based immigrant schemes for GAs to deal with DOPs [6]. The former scheme aims at improving the performance on GAs in slowly and slightly changing environments while the latter scheme makes GAs adapted to more severely changing environments. As examples of the *indirect immigrant scheme*, Yang [18] used a memory as the model to generate immigrants. Besides, in ref. [4], a vector with the allele distribution of the population was first calculated and then was used to generate immigrants. Moreover, experiments were done to thoroughly investigate the difference in the behaviors of different types of immigrant schemes and to compare their performance in dynamic environments [3]. The authors also proposed a hybrid immigrant scheme which combined the merits of the two types of immigrant schemes.

### 3 Adapting the replacement rate

The replacement rate was claimed to be an important issue needed to be considered when designing immigrant schemes for EAs in dynamic environments [3]. Grefenstette [16] conducted an experiment showing that for the non-stationary test function in his study, random immigrant schemes for GAs using a replacement rate 0.3 exhibited the best off-line performance. After that, all the immigrant schemes developed so far adopted 0.2 or 0.3 as the fixed replacement rate throughout the run [3 – 6, 16]. However, we believe that a fixed replacement rate is not the best choice, since for different immigrant schemes, at different stages of evolutionary process, and under different environments, the most appropriate replacement rate varies. It is natural that the replacement rate should be set to a proper value such that a good balance between exploration and exploitation can be kept. If the immigrant scheme has a positive effect, i.e., the introduced immigrants can help the population to move to the new optimum when an environmental change occurs, or can speed up the convergence of the population when there is no change, the replacement rate should be enlarged. Otherwise, to prevent the searching process of EAs from being disrupted, the replacement rate should be reduced.

In order to adapt the value of the replacement rate during the searching process, we must estimate the effect of the current immigrant scheme. If the current immigrant scheme has a positive effect, more immigrants are encouraged to be introduced, i.e., the value of the replacement rate should be enlarged. Otherwise, the value of the replacement rate should be reduced. We propose to monitor the effect of an immigrant scheme as below.

#### 3.1 Evaluating the effect of immigrant schemes

Denote the number of immigrants at time  $t$  by  $N\_Im(t)$ , the fitness value of the  $i$ th immigrant at time  $t$  by  $f_{im_i}(t)$  ( $i = 1, 2, \dots, N\_Im(t)$ ), the median fitness value of the population at time  $t$  by  $f_{median}(t)$ , and the number of immigrants satisfying the condition  $f_{im_i}(t) \geq f_{median}(t)$  at time  $t$  by  $N\_Im(t)\{f_{im_i}(t) \geq f_{median}(t)\}$ . Then the effect of an immigrant scheme at time  $t$  is denoted by  $Eff(t)$  and can be evaluated as follows:

$$Eff(t) = \frac{N\_Im(t)\{f_{im_i}(t) \geq f_{median}(t)\}}{N\_Im(t)}. \quad (1)$$

Given a predefined threshold  $\delta_{eff}$ , the immigrant scheme can be regarded as having a positive effect at time  $t$  if  $Eff(t) > \delta_{eff}$ . Note that in this paper, the time  $t$  is measured in generations. Instead of the median fitness value of the population used in Eq. (1), the best or the worst fitness values of the population can also be used. We choose “median” because it is not too extreme.

It can be seen from the Eq. (1) that we first count the number of immigrants that are regarded as being fit according to the results of comparing with a certain criterion derived from the current population, and then we can measure the effect of an immigrant scheme by calculating the proportion of the obtained

number in the whole population of the immigrants. Naturally, this measure is by no means the most objective, but it can to some extent reflect the effect of the immigrant scheme. Therefore, we design this measure for the self-adaptive scheme described below.

### 3.2 Adapting the replacement rate

As mentioned above, if the current immigrant scheme has a positive effect, the replacement rate should be increased, and otherwise it should be decreased. Therefore, the replacement rate at time  $t$  (denoted by  $R(t)$ ) can be derived from  $R(t-1)$  according to the effect of the immigrant scheme at time  $t-1$ . Given the effect of immigrant scheme at time  $t-1$ , we have the following replacement rate updating rule:

$$R(t) = \begin{cases} R(t-1) + 0.1, & \text{if } Eff(t-1) > \delta_{eff} \\ R(t-1) - 0.1, & \text{if } Eff(t-1) < \delta_{eff} \\ R(t-1), & \text{otherwise.} \end{cases} \quad (2)$$

$R(t)$  is bounded in the interval  $(0, 1)$ , and  $R(0) = 0.5$ . It can be seen that our approach introduces a new parameter  $\delta_{eff}$ . However, we will show that the performance is much less sensitive to the value of  $\delta_{eff}$  than to the replacement rate as in the immigrant scheme without the adaptation mechanism.

### 3.3 Time complexity of the adaptation mechanism

The additional cost added by the adaptation process is determined by the calculation of  $Eff(t)$  in Eq. (1) and the update of  $R(t)$  in Eq. (2). When computing the runtime complexity, since data transfer operations like copy/assignment, etc. hardly require any complex digital circuitry like adder, comparator, etc. [20, 21], we only consider the fundamental floating-point arithmetic and logical operations performed by the adaptation mechanism. Given the population of size  $N$  and at time  $t$ , the complexity  $T(N)$  of the adaptation consists of finding the median of the population, calculating  $Eff(t-1)$ , and updating  $R(t)$ . Based on the partition algorithm similar to the one used in quicksort, finding the median needs  $O(N)$  comparisons on average and  $O(N^2)$  comparisons in the worst case [20]. Calculating  $Eff(t-1)$  needs  $R(t-1) \cdot N$  comparisons, and updating  $R(t)$  needs one comparison and one arithmetic operation (when  $Eff(t) \neq \delta_{eff}$ ). Therefore, the time complexity of the adaptation mechanism  $T(N) = O(N)$  on average and  $T(N) = O(N^2)$  in the worst case.

In real-world applications, the evaluation of an individual may consume too much time. Therefore, given a population of size  $N$ , in one generation, compared with  $N$  times of evaluation, the time consumed by floating-point arithmetic and logical operations introduced by the adaptation process is negligible.

## 4 Experimental design

### 4.1 Dynamic test environments

Using the DOP generator proposed in refs. [22, 23], dynamic environments can be constructed from any binary-encoded stationary function  $f(\mathbf{x})$  ( $\mathbf{x} \in \{0, 1\}^l$ ) by a bitwise exclusive-or (XOR) operator. The generator basically works as follows: The operation  $\mathbf{x} \oplus \mathbf{M}$  is performed before evaluation of the individual  $\mathbf{x}$  in the population, where  $\oplus$  is the XOR operator (i.e.,  $1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 0 = 0$ ) and  $\mathbf{M}$  is a previously created binary mask. Then the resulting individual is evaluated. When there is a change at generation  $t$ , we have  $f(\mathbf{x}, t+1) = f(\mathbf{x} \oplus \mathbf{M}, t)$ . With this DOP generator, the speed and the severity of the environmental changes are controlled by parameters  $\tau$  and  $\rho \in (0.0, 1.0)$  respectively. The parameter  $\tau$  is the number of generations between two changes, and the parameter  $\rho$  is the ratio of ones in an intermediate binary template that is created for each environment and used to generate the mask  $\mathbf{M}$ . Smaller  $\tau$  means faster changes while bigger  $\rho$  indicates severer changes. More details about the DOP generator can be found in refs. [22, 23].

In this paper, dynamic test environments are systematically constructed from a stationary problems, i.e., the OneMax problem, via the DOP generator with  $\tau$  set to 10, 50, and 100 and  $\rho$  set to 0.05, 0.5,

and 0.9 respectively. In all, a series of 9 DOPs are generated. The indices of the environment used in the experiment are shown in Table 1.

**Table 1** Environmental indices used in the experiment

Environmental Parameters	$\tau = 10$	$\tau = 50$	$\tau = 100$
$\rho = 0.1$	1	2	3
$\rho = 0.5$	4	5	6
$\rho = 0.9$	7	8	9

#### 4.1.1 The OneMax problem

The OneMax problem has only one optimum and aims to maximize the number of ones in a binary string. So the fitness of an individual is the number of ones in the binary string. Given a binary string  $\mathbf{x}$  of length  $L$ , the problem is defined as follows:

$$\text{maximize } f(\mathbf{x}) = \sum_{i=1}^L x_i, \quad (3)$$

where  $f(\mathbf{x})$  is the fitness of a binary string  $\mathbf{x} = (x_1, \dots, x_L) \in I = \{0, 1\}^L$ . In our experiment we set the length of the binary string  $L$  to 100.

#### 4.1.2 The Royal Road problem

The Royal Road function is a binary problem with only one optimum and many large plateaus. The function used in this paper is similar to the Royal Road function introduced in ref. [24]. It is defined on a 100-bit binary string that consists of 25 contiguous building blocks, each of which is 4-bit long and contributes  $c_i = 4$  ( $i = 1, \dots, 25$ ) to the total fitness if and only if every bit is one. Therefore, the fitness of a string  $\mathbf{x}$  is the sum of the coefficients  $c_i$  corresponding to each given schema  $s_i$ , of which  $\mathbf{x} \in s_i$ , i.e.:

$$\text{maximize } f(\mathbf{x}) = \sum_{i=1}^{25} c_i \delta_i(\mathbf{x}), \quad (4)$$

where

$$\delta_i(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in s_i \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

## 4.2 Experimental setup

In order to investigate the efficacy of the proposed self-adaptive mechanism for immigrant schemes, we applied it to two previously proposed immigrant schemes for GAs, i.e., the random immigrant scheme GA (RIGA) [16] and the elitism-based immigrant scheme GA (EIGA) [5]. In each generation, RIGA generates immigrants randomly, and EIGA generates immigrants by mutating the elite of the population at previous generation. These immigrant schemes with self-adaptive replacement rates are denoted by SRRIGA and SREIGA (“SR” stands for “self-adaptive rate”), respectively in this paper. RIGA and SRRIGA are the same except that the former has a fixed replacement rate while the latter has the self-adaptive replacement rate. The same relation exists between EIGA and SREIGA. The pseudo-codes of the algorithms are given in Table 2, where  $p^c$  is the crossover probability,  $p^m$  is the mutation probability, and  $p_{ei}^m$  is the probability of mutating the elite, and  $n$  is the population size.

The algorithms’ genetic operations and parameters were set as follows: generational, uniform crossover with crossover probability  $p^c = 0.6$ , bit-flipping mutation with mutation rate  $p^m = 0.01$ , fitness proportional selection implemented via stochastic universal sampling algorithm with elitism of size 1, chromosome length  $L = 100$ , and population size  $n = 100$ . For EIGA,  $p_{ei}^m$  was set to 0.01. For RIGA and EIGA, in order to study the impact of replacement rate on the performance of them in dynamic environments, for each algorithm the replacement rate was set from 0.1 to 0.9 with step 0.1. For SRRIGA and SREIGA, to investigate the impact of  $\delta_{eff}$  on the performance of algorithms, for each algorithm the value  $\delta_{eff}$  was set from 0.1 to 0.9 with step 0.1.

**Table 2** Pseudo-code for GAs with random immigrants (RIGA), with elitism-based immigrants (EIGA), and their self-adaptive variants (SRRIGA and SREIGA)

```

1: begin
2:    $t := 0$  and initialize population  $P(0)$  randomly
3:   evaluate population  $P(0)$ 
4:   if self-adaptation is used then
// for SRRIGA and SREIGA
5:      $R(0) := 0.5$ 
6:   else
7:     set  $R(0)$  to a predefined value
8:   end if
9:   repeat
10:     $P'(t) := \text{selectForReproduction}(P(t))$ 
11:     $\text{crossover}(P'(t), p^c)$  //  $p^c$  is the crossover probability
12:     $\text{mutate}(P'(t), p^m)$  //  $p^m$  is the mutation probability
13:    evaluate population  $P'(t)$ 
14:    if random immigrant scheme is used then
// for RIGA
15:      generate  $R(t) \times n$  random immigrants
16:      evaluate these random immigrants
17:    end if
18:    if elitism-based immigrant scheme is used then
// for EIGA
19:      denote the elite in  $P(t-1)$  by  $E(t-1)$ 
20:      generate  $R(t) \times n$  immigrants by mutating  $E(t-1)$ 
with  $p_{ei}^m$ 
21:      evaluate these elitism-based immigrants
22:    end if
23:    if self-adaptation is used then
// for SRRIGA and SREIGA
24:      calculate  $\text{Eff}(t)$  according to Eq. (1)
25:      update  $R(t)$  according to Eq. (2)
26:    end if
27:    replace the worst individuals in  $P'(t)$  with the
generated immigrants
28:     $P(t+1) := P'(t)$ 
29:  until the termination condition is met
30: end

```

For each algorithm on a DOP, 30 independent runs were executed with the same set of random seeds, and for each run, 50 environmental changes were allowed. The off-line performance used to compare the algorithms was defined as below:

$$\bar{F}_{BOG} = \frac{1}{G} \sum_{i=1}^G \left( \frac{1}{N} \sum_{j=1}^N F_{BOG_{ij}} \right), \quad (6)$$

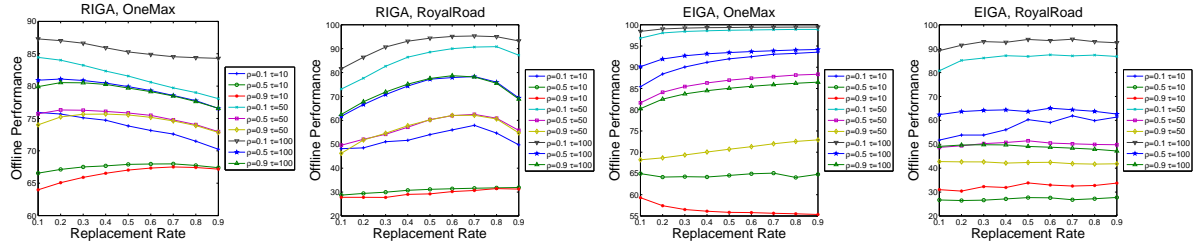
where  $G$  is the number of generations for a run,  $N$  is the number of runs, and  $F_{BOG_{ij}}$  is the best-of-generation fitness of generation  $i$  of run  $j$ .

In the experiments, we would like to investigate the impact of the replacement rate on the performance of algorithms, the impact of the newly introduced parameter  $\delta_{eff}$  on the performance of algorithms, and compare the performance of algorithms with and without the adaptation. The results and the related analyses are shown in the next section.

## 5 Experimental results

### 5.1 The impact of replacement rate on the performance of algorithms

The off-line performance of RIGA and EIGA on all DOPs against the replacement rate is plotted in Figure 1. From Figure 1, it can be seen that for RIGA on the OneMax problem, when the environment changes slightly, i.e.,  $\rho = 0.1$ , smaller replacement rates always show higher performance than larger replacement rates. This is because when the environment changes slightly, the individuals in the population are still quite fit in the new environment, and the introduced randomly created individuals will disrupt the ongoing searching process, leading to the degradation in the performance. Besides, for other kinds of



**Figure 1** The impact of the replacement rate on the performance of algorithms in dynamic environments.

environments on the OneMax problem, i.e.,  $\rho = 0.5$  and  $\rho = 0.9$ , and all kinds of environments except  $\rho = 0.5, 0.9$  and  $\tau = 10$  on the Royal Road problem, the performance first increases and then decreases as the value of replacement rate increases. The reason is that these kinds of environments are characterized as more severe degree of changes, so neither must the value of replacement rate be too small in order to help the population to jump out of the old promising region, nor must the value of replacement rate be too large in order to protect the searching process from being disrupted.

For EIGA, it can be seen from Figure 1 that on the OneMax problem, except when  $\tau = 10$  and  $\rho = 0.5, 0.9$ , larger replacement rates always exhibit higher performance than smaller replacement rates. This is because EIGA is especially designed for addressing slightly and slowly changing environments. When the environment changes slightly, i.e.,  $\rho = 0.1$ , since the new optimum is not far from the old one, elitism-based immigrants can be quite fit in the new environment. On the other hand, when the environment changes slowly, i.e.,  $\tau = 50$  and  $100$ , there is enough time for the population to recover from the last change of the environment, and to find the new optimum quickly with the help of elitism-based immigrants even when the environment changes significantly (i.e.,  $\rho = 0.9$ ). However, when the environment changes rapidly, i.e.,  $\tau = 10$ , there is no time for the population to quickly recover under more severe changes (i.e.,  $\rho = 0.5$  and  $0.9$ ). Therefore, increasing the value of replacement rate does not improve the performance under this situation. In addition, on the Royal Road problem, increasing the replacement rate brings a little performance enhancement for most slightly and slowly changing environments, and sometimes there is even a little degradation in the performance. This is because by mutating the elite, some good building blocks may be destroyed.

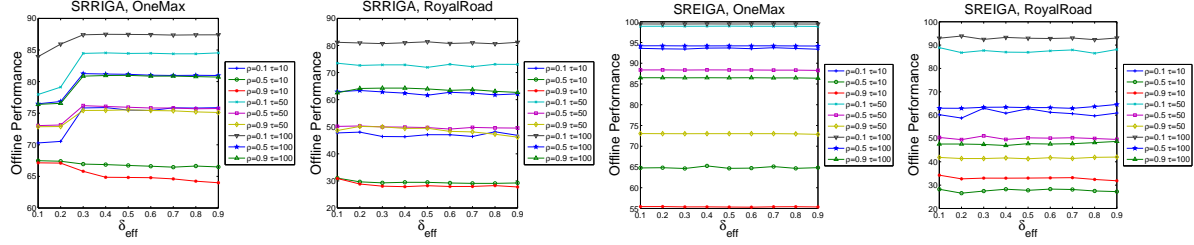
In summary, the experimental results validate our claim in Section 3 that for different immigrant schemes, at different stages of evolutionary process, and under different environments, there exist different replacement rates to maximize the performance.

## 5.2 The impact of $\delta_{eff}$ on the performance of algorithms

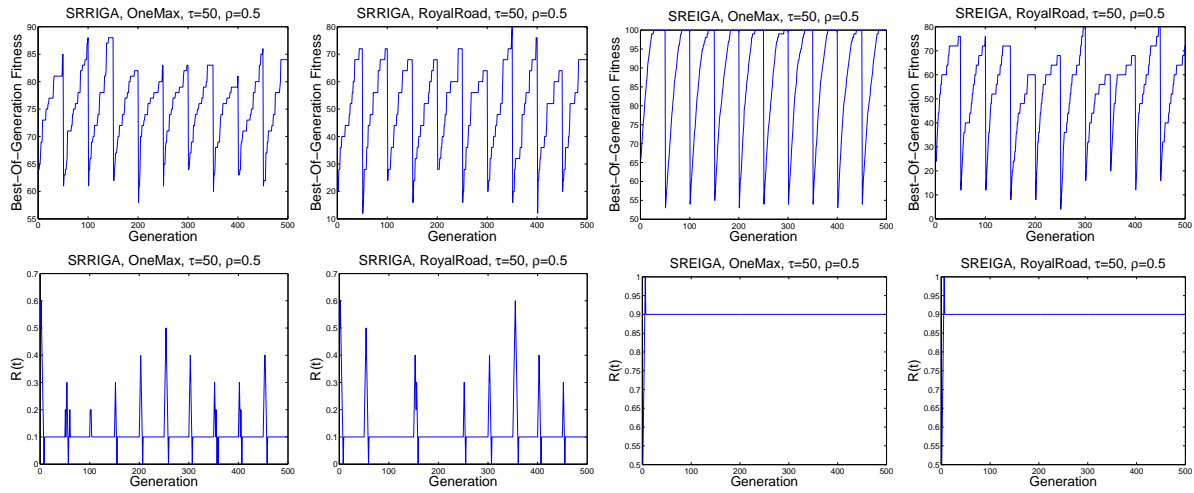
The off-line performance of SRRIGA and SREIGA on all DOPs against the value of  $\delta_{eff}$  is plotted in Figure 2. From this figure, several results can be observed and analyzed as follows.

First, it can be seen from Figure 2 that on the OneMax problem, when the environment changes slightly (i.e.,  $\rho = 0.1$ ) or slowly (i.e.,  $\tau = 50$  and  $100$ ), the performance of SRRIGA increases while  $\delta_{eff}$  increases from 0.1 to 0.3, and then maintains nearly the same level when  $\delta_{eff}$  is equal to or larger than 0.3. On the other hand, when the environment changes both significantly and rapidly, the performance of SRRIGA keeps a certain level when  $\delta_{eff}$  is not larger than 0.2 and decreases otherwise. This can be explained as follows.

Through the definition of  $Eff(t)$  in Eq. (1) and the meaning of  $\delta_{eff}$ , we know that if  $\delta_{eff}$  is too small, the introduced immigrants will easily tend to be regarded as having a positive effect, and thus according to Eq. (2), the value of replacement rate will increase. On the other hand, if  $\delta_{eff}$  is too large, the immigrants introduced will mostly be regarded as having a negative effect, causing the reduction of the value of replacement rate. Meanwhile, from subsection 5.1, we know that immigrants in SRRIGA are useful for rapidly and significantly changing environments, and are harmful for other kinds of environments if their number is too large. Therefore, in slightly or slowly changing environments, the performance when



**Figure 2** The impact of the value of  $\delta_{eff}$  on the performance of algorithms in dynamic environments.



**Figure 3** The dynamics of the maximal fitness and the replacement rate of a run ( $\delta = 0.3$ ) in the first 10 environments ( $\tau = 50$  and  $\rho = 0.5$ ).

$\delta_{eff}$  is too small (e.g.,  $\delta_{eff} = 0.1$ ) is worse than the performance when  $\delta_{eff}$  is larger (e.g.,  $\delta_{eff} \geq 0.3$ ), while in significantly and rapidly changing environments, the performance will degrade when  $\delta_{eff} > 0.2$ .

Second, the performance of SRRIGA on the Royal Road problem generally does not rely on  $\delta_{eff}$ , which is due to the fact that random immigrants may introduce useful schemata, and at the same time the harmful schemata which can break good building blocks via crossover. On the other hand, the performance of SREIGA on both problems is generally insensitive to  $\delta_{eff}$ . This is because immigrants in SREIGA are always useful in a stationary period between two consecutive changes since they can speed up the convergence rate. Therefore, no matter how much  $\delta_{eff}$  is, immigrants in SREIGA will have a positive effect in most time, see Figure 3. Moreover, from Figure 3, on the OneMax problem, we can see that the replacement rate increased after the environmental changes, which indicates that the random immigrants at that time were useful for the population to move to the new optimum, and then the it decreased which means the promising area was found and too many immigrants could disrupt the ongoing search.

To sum up, the performance of the algorithm with adaptation of the replacement rate is not so sensitive to  $\delta_{eff}$  as that of the algorithm without adaptation to the value of the replacement rate. More importantly, there exist some values of  $\delta_{eff}$  (e.g.,  $\delta_{eff} \in [0.3, 0.9]$ ) that can yield satisfying performance for every immigrant scheme in different kinds of environments experimented. Therefore, the tedious work of fine-tuning the value of replacement rate in algorithms without adaptation in order to maximize the performance for a certain immigrant scheme and for a certain environment can be avoided.

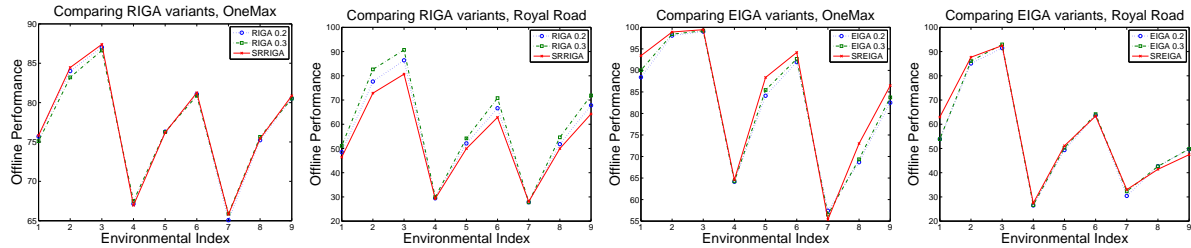


Figure 4 Comparison between immigrant schemes with and without adaptation mechanism.

### 5.3 Adaptation versus no adaptation

The experimental results regarding overall performance are presented in Table 3, where the performance of the algorithm adopting adaptation scheme when  $\delta_{eff} = 0.3$  is compared with the performance of the algorithm with a fixed replacement rate. We did experiments with the replacement rate being from 0.1 to 0.9 with step size 0.1. Here, only the results obtained with the commonly used values 0.2 and 0.3 for the fixed replacement rate scheme are presented in Table 3. The results are also given in Figure 4. Furthermore, the best results achieved with the fixed replacement rate are also presented, together with the corresponding replacement rate (the column “*best(R)*”). These three sets of results were compared to the results obtained by the self-adaptive replacement rate. Wilcoxon rank sum tests at a 0.05 level of significance were used to check whether there existed significant difference between fixed and self-adaptive replacement rates. The results of statistical tests are also presented in Table 3, where the result is marked as “+” or “-” when the algorithm with adaptation scheme ( $\delta_{eff} = 0.3$ ) is significantly better or worse than the algorithm without adaptation scheme, respectively, and “~” indicates there is no statistical difference between two algorithms. Analyzing the results in Table 3 and Figure 4, several phenomena can be observed and can be explained as below.

First, comparing the self-adaptive rate with commonly used replacement rates, we can see that on the OneMax problem applying adaptation to EIGA improves its performance a lot except when  $\rho = 0.9$  and  $\tau = 10$ . This is because when the environment changes both severely and rapidly, immigrants generated via mutating the elite from previous generation usually have low fitness in the new environment. For RIGA on the other hand, the difference between the performance of the algorithm before and after using the adaptation scheme is not so obvious. A probable reason is that the stochasticity in RIGA can sometimes mislead the adaptation of the replacement rate. On the Royal Road problem, due to the reason described in subsection 5.2, the performance enhancement of SREIGA is not obvious, and the performance of SRRIGA is the worst.

Second, comparing the performance using self-adaptive rate with the best results obtained using fixed replacement rate (the column “*best(R)*”), the former is beaten by the latter in most cases. This can be expected since the results shown in the column “*best(R)*” are obtained via fine-tuning the replacement rate. Fortunately, except for SRRIGA on the Royal Road problem, even if the algorithm with adaptation when  $\delta_{eff} = 0.3$  does not show the best score, its performance level is still quite satisfactory. Considering the tedious work of fine-tuning the replacement rate for every immigrant scheme and for every environment, it is worthy using our adaptation scheme and setting the value of  $\delta_{eff}$  simply to 0.3. The very slight performance degradation when using the adaptive scheme is more than compensated for by the convenience of not having to set the replacement rate by hand.

## 6 Conclusions

This paper examined the impact of the replacement rate on the performance of EAs with immigrants schemes in dynamic environments. A self-adaptive replacement rate was proposed to address DOPs. Experimental studies were carried out on a series of dynamic problems to investigate the efficacy of our approach. From the obtained results several conclusions can be drawn: First, the replacement rate is an

important issue when designing immigrant schemes for EAs to cope with DOPs, and the most appropriate choice of the replacement rate varies according to the problems being solved, environments encountered, mechanisms used to generate immigrants, and stages at which the search is going. Second, comparing with immigrant schemes with a fixed replacement rate adopting traditionally used values (i.e., 0.2 and 0.3), our proposed adaptation scheme can promote the performance of immigrants schemes in most cases. Finally, compared with tedious work of fine-tuning the replacement rate manually, the proposed approach is more convenient and can produce sufficiently good solutions under different conditions. However, to pay a price for this convenience, the performance decreases a little in some cases. As to the future work, some other ways of adapting the replacement rate are under investigation.

**Table 3** Experimental results regarding overall performance and results of comparing the performance of algorithms via

Adaptation vs No Adaptation	statistical tests														
	OneMax Results				Statistical test results				Royal Road Results						
GAs and Parameters	Overall performance		$\delta_{eff} = 0.3$		vs 0.2 vs 0.3 vs $best(R)$			Overall performance		Statistical test results					
	0.2	0.3	$\delta_{eff} = 0.3$	$best(R)$	vs 0.2	vs 0.3	vs $best(R)$	0.2	0.3	$\delta_{eff} = 0.3$	$best(R)$	vs 0.2	vs 0.3	vs $best(R)$	
$\rho = 0.1$	$\tau = 10$														
	RIGA	75.69	75.11	75.83	<b>75.93</b> (0.1)	+	+	-	48.50	51.06	46.38	<b>57.91</b> (0.7)	-	-	-
	EIGA	88.41	90.05	93.41	<b>93.54</b> (0.9)	+	+	-	53.85	53.84	<b>62.95</b>	61.89(0.7)	+	+	+
	$\tau = 50$														
	RIGA	84.01	83.21	<b>84.46</b>	84.44(0.1)	+	+	+	77.63	82.64	72.84	<b>90.84</b> (0.8)	-	-	-
	EIGA	98.08	98.42	<b>98.91</b>	98.90(0.8)	+	+	~	85.07	86.07	<b>87.67</b>	87.42(0.6)	+	+	+
$\tau = 100$															
RIGA	87.02	86.63	<b>87.40</b>	87.29(0.1)	+	+	+	86.39	90.69	80.69	<b>95.31</b> (0.7)	-	-	-	
EIGA	99.01	99.20	<b>99.46</b>	<b>99.46</b> (0.9)	+	+	~	91.43	92.95	92.44	<b>93.95</b> (0.7)	+	~	-	
$\rho = 0.5$	$\tau = 10$														
	RIGA	67.13	67.51	66.95	<b>67.99</b> (0.7)	-	-	-	29.44	29.96	29.28	<b>31.91</b> (0.9)	~	~	-
	EIGA	64.15	64.25	64.64	<b>65.08</b> (0.7)	+	+	-	26.46	26.61	27.46	<b>27.74</b> (0.9)	+	+	-
	$\tau = 50$														
	RIGA	<b>76.31</b>	76.27	76.19	<b>76.31</b> (0.2)	-	-	-	52.09	54.17	49.94	<b>62.51</b> (0.7)	-	-	-
	EIGA	84.11	85.45	<b>88.37</b>	88.34(0.9)	+	+	+	49.36	50.26	51.10	<b>51.45</b> (0.5)	+	+	-
$\tau = 100$															
RIGA	81.09	80.87	<b>81.26</b>	81.09(0.2)	+	+	+	66.61	70.80	62.85	<b>78.32</b> (0.7)	~	-	-	
EIGA	91.94	92.69	<b>94.18</b>	94.17(0.9)	+	+	~	63.68	64.18	63.36	<b>65.09</b> (0.6)	~	-	-	
$\rho = 0.9$	$\tau = 10$														
	RIGA	65.07	65.90	65.80	<b>67.51</b> (0.7)	+	-	-	27.80	27.77	28.06	<b>31.46</b> (0.8)	+	+	-
	EIGA	57.42	56.52	55.38	<b>59.30</b> (0.1)	-	-	-	30.42	32.29	33.00	<b>33.84</b> (0.5)	+	+	-
	$\tau = 50$														
	RIGA	75.23	75.65	75.43	<b>75.68</b> (0.4)	+	-	-	51.77	54.60	49.95	<b>62.12</b> (0.7)	-	-	-
	EIGA	68.68	69.36	<b>73.01</b>	72.90(0.9)	+	+	+	42.65	42.62	41.39	<b>42.78</b> (0.1)	~	~	-
$\tau = 100$															
RIGA	80.54	80.51	<b>80.87</b>	80.54(0.2)	+	+	+	67.78	71.87	64.26	<b>78.69</b> (0.6)	-	-	-	
EIGA	82.49	83.73	86.48	<b>86.49</b> (0.9)	+	+	~	49.68	49.80	47.43	<b>49.80</b> (0.3)	-	-	-	

## Acknowledgements

This work was partially supported by Natural Science Foundation of China grant (No. U0835002), the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033), an EPSRC grant (EP/E058884/1) on ‘‘Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications’’, and the Fund for Creative Research for Graduate Students of University of Science and Technology of China.

## References

- 1 Jin Y, Branke J. Evolutionary optimization in uncertain environments: a survey. *IEEE Transactions on Evolutionary Computation*, 2005, 9(3): 303-317
- 2 Rohlfschagen P, Yao X. Attributes of dynamic combinatorial optimisation. In: *Proceedings of the 7th International Conference on Simulated Evolution And Learning (SEAL'2008)*, LNCS Vol 5361, Springer-Verlag, Berlin, 2008, 442-451
- 3 Yu X, Tang K, Chen T, Yao X. Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing*, 2009, 1(1): 3-24
- 4 Yu X, Tang K, Yao X. An immigrants scheme based on environmental information for genetic algorithms in changing environments. In: *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*. 2008, 1141-1147
- 5 Yang S. Genetic algorithms with elitism-based immigrants for changing optimization problems. In: *Applications of Evolutionary Computing 2007*, LNCS Vol 4448. Springer-Verlag, Berlin Heidelberg 2007. 627-636

- 6 Yang S, Tinós R. A hybrid immigrants scheme for genetic algorithms in dynamic environments. *International Journal of Automation and Computing*, 2007, 4(3): 243-254
- 7 Tinós R, Yang S. Genetic algorithms with self-organized criticality for dynamic optimization problems. In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*. 2005, Vol 3, 2816-2823
- 8 Hinterding R, Michalewicz Z, Eiben A E. Adaptation in evolutionary computation: a survey. In: *Proceedings of the 4th International Conference on Evolutionary Computation*. 1997, 65-69
- 9 Shi J P, Zhang W G, Li G W, Liu X X. Research on allocation efficiency of the redistributed pseudo inverse algorithm. *Science in China Series F: Information Sciences*. 2010, 53(2):271-277
- 10 Zhang J, Lo W L, Chung H. Pseudo-coevolutionary genetic algorithms for power electronics regulators optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*. 2006, 36(4): 590-598
- 11 Zhang J, Chung 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 Yao X, Liu Y, Lin G. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*. 1999, 3(2): 82-102
- 13 Beyer H-G. Toward a theory of evolution strategies: self-adaptation. *Evolutionary Computation*. 1996, 3(3): 311-347
- 14 Qin A K, Liang 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
- 15 Zhan Z H, Zhang J, Li Y, Chung H. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. 2009, 39(6): 1362-1381
- 16 Grefenstette J J. Genetic algorithms for changing environments. In: *Proceedings of Parallel Problem Solving from Nature Vol 2*. Elsevier Science Publishers, The Netherlands, 1992, 137-144
- 17 Cobb H G, Grefenstette J J. Genetic algorithms for tracking changing environments. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. 1993, 523-530
- 18 Yang S. Memory-based immigrants for genetic algorithms in dynamic environments. In: *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*. 2005, Vol 2, 1115-1122
- 19 Branke J. Memory enhanced evolutionary algorithms for changing optimization problems. In: *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*. 1999, Vol 3, 1875-1882
- 20 Cormen T H, Leiserson C E, Rivest R L, Stein C. *Introduction to algorithms*, 2nd ed. Cambridge, MA: MIT Press and New York: McGraw-Hill, 2001
- 21 Aho A V, Hopcroft J E, Ullman J D. *Data structures and algorithms*. Reading, MA: Addison-Wesley, 1983
- 22 Yang S. Non-stationary problem optimization using the primal-dual genetic algorithm. In: *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*. 2003, Vol 3, 2246-2253
- 23 Yang S, Yao X. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 2005, 9(11): 815-834
- 24 Mitchell M, Forrest S, Holland J H. The royal road for genetic algorithms: fitness landscapes and GA performance. In: *Proceedings of the 1st European conference on artificial life*. MIT Press, Cambridge, MA, USA. 1991, 245-254