Memetic Comp. (2009) 1:3–24
DOI 10.1007/s12293-008-0003-6

REGULAR RESEARCH PAPER

# Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization

**Xin Yu · Ke Tang · Tianshi Chen · Xin Yao**

Received: 16 August 2008 / Accepted: 03 October 2008 / Published online: 12 December 2008


**Abstract** In recent years, there has been a growing interest in studying evolutionary algorithms (EAs) for dynamic optimization problems (DOPs). Among approaches developed for EAs to deal with DOPs, immigrants schemes have been proven to be beneficial. Immigrants schemes for EAs on DOPs aim at maintaining the diversity of the population throughout the run via introducing new individuals into the current population. In this paper, we carefully examine the mechanism of generating immigrants, which is the most important issue among immigrants schemes for EAs in dynamic environments. We divide existing immigrants schemes into two types, namely the *direct immigrants scheme* and the *indirect immigrants scheme*, according to the way in which immigrants are generated. Then experiments are conducted to understand the difference in the behaviors of different types of immigrants schemes and to compare their performance in dynamic environments. Furthermore, a new immigrants scheme is proposed to combine the merits of two types of immigrants schemes. The experimental results show that the interactions between the two types of schemes reveal positive effect in improving the performance of EAs in dynamic environments.

X. Yu · K. Tang (✉) · T. Chen · X. Yao
Nature Inspired Computation and Applications Laboratory,
Department of Computer Science and Technology,
University of Science and Technology of China,
Hefei, Anhui 230027, China
e-mail: ketang@ustc.edu.cn

X. Yu
e-mail: hughyx@mail.ustc.edu.cn

T. Chen
e-mail: cetacy@mail.ustc.edu.cn

X. Yao
CERCIA, The School of Computer Science,
University of Birmingham, Edgbaston,
Birmingham B15 2TT, UK
e-mail: x.yao@cs.bham.ac.uk

**Keywords** Evolutionary Algorithm · Dynamic Optimization Problem · Immigrants scheme

## 1 Introduction

It is well-known that evolutionary algorithms (EAs) are powerful techniques for solving various kinds of optimization problems in real-world applications [1,2]. Traditionally, researchers have been concentrating their attentions on EAs applied to stationary optimization problems, where problems are given in advance and maintain fixed during the evolutionary process. However, in many real-world applications, we have to deal with dynamic optimization problems (DOPs) [3]. In DOPs, the environment, including the objective function, the decision variables, the problem instance, constraints and so on, may vary over time. When the changes take place, it may take some time for the EA to adapt to the new environment. Due to this characteristic of DOPs, the EAs designed specifically for the stationary optimization problems, in which the environment will not change at all, may no longer be efficient.

Recently, developing EAs for DOPs has attracted a growing interest due to its importance in real-world applications [3,4]. The simplest strategy to cope with a change of the environment is to regard every change as the arrival of a new optimization problem that has to be solved from scratch [5]. However, this strategy is undesirable because it generally requires substantial computational efforts. Thus, more complicated strategies are required to reduce the computational efforts and maintain a high quality of the output solutions at the same time. Over the years, several specific strategies have been proposed for EAs on DOPs, including diversity reinforcing or maintaining schemes [6–11], memory schemes [12–19], multi-population schemes [20–22], and their hybridizations [23,24]. Some comprehensive surveys can be found in the books by Branke [25], Morrison [26], and Weicker [27]. There is also a frequently updated online repository [28].

As we have seen, many approaches for EAs on DOPs concentrate on maintaining or reinforcing the diversity of EAs, among which, immigrants schemes are the simplest approaches to implement and have been validated to be efficient [8,24,29,21,30–32]. Immigrants schemes attempt to maintain the diversity of the population via introducing new individuals into the current population. In this paper, compared with works in literature, the mechanism of generating immigrants, which is the most important issue of the immigrants schemes, is examined intensively. According to the way in which immigrants are created, we categorize the existing immigrants schemes into two types, i.e., the *direct immigrants scheme* and the *indirect immigrants scheme*. Then, an experimental study is carried out on several immigrants schemes to analyze the behaviors of the two types of immigrants schemes. Based on the analysis, a new immigrant scheme is proposed, which tries to combine the merit of the *direct immigrants scheme* and the merit of the *indirect immigrants scheme*. Experimental results validate the benefit of the proposed immigrants scheme for EAs in dynamic environments.

The rest of this paper is outlined as follows. Section 2 examines design decisions related to immigrants schemes. Section 3 describes measures of performance to understand the behavior of EAs in dynamic environments. Section 4 details some EAs investigated in this paper including the EA with the newly proposed immigrants scheme. Section 5 briefly reviews existing DOP generators and then presents the dynamic test environments constructed for the experimental study in this paper. Section 6 presents the experimental results and analysis. Finally, Sect. 7 concludes this paper with discussions on relevant future work.

## 2 Immigrants schemes for EAs in dynamic environments

When addressing DOPs, traditional EAs cannot adapt well to the new environment when changes occur once converged. The application of immigrants schemes has proved to be able to enhance the performance of EAs in dynamic environments [8,24,29,21,30–32]. The basic principle of immigrants schemes is to introduce new individuals into the evolving population to replace a predefined portion of the population, and thus, the diversity of the population can be maintained throughout the run. Immigrants schemes mainly involve four concerns: how to generate immigrants, how to set the number of immigrants, how to design replacement strategy, and how to boost the survival probability of the newly introduced immigrants [32].

For the first concern, in order to increase the diversity of the population, the most prominent approach to generate immigrants seems to create immigrants randomly. Grefenstette [8] used randomly generated individuals to replace the worst individuals of the population in each generation. The random immigrants scheme works on the analogy of the flux of immigrants that wander in and out of a population between two generations in nature [29]. His study shows that the random immigrants scheme works well in environments where there are occasional, large changes in the location of the optimum.

As has been argued by Branke [13], continuous adaption only makes sense when problems to be studied feature “small to medium” environmental changes, otherwise to restart the search from scratch would be the proper choice. Under this presupposition, random immigrants scheme may not be suitable, since random immigrants may actually be of little use when individuals in the previous environment may still be quite fit in the new environments. To handle these problems, the information of the population can be used to help to generate immigrants. It is hoped that the information guided immigrants would be more adapted to different kinds of environmental changes.

The approaches of utilizing the information of the population can be categorized into two types, i.e., direct approaches and indirect approaches, which produce the *direct immigrants scheme* and the *indirect immigrants scheme*, respectively. The *direct immigrants scheme* generates immigrants based on the current population. Examples are elitism-based immigrants scheme [30] in which immigrants come from mutating the elite from previous generation, and a hybrid immigrants scheme combing the elitism-based, the traditional random, and the dualism-based immigrants schemes for GAs to deal with DOPs [31]. The former scheme aims to improve the performance on GAs in slowly and slightly changing environments while the latter scheme makes GAs adapted to more severely changing environments. On the other hand, the *indirect immigrants scheme* first builds a model based on the current population, then generates immigrants according to the model. For example, in [24], a memory was used as the model to generate immigrants. Besides, in [32], a vector with the allele distribution of the population was first calculated and then was used to generate immigrants for GAs to address DOPs with some preliminary results.

As to the number of immigrants, in order to prevent immigrants from disrupting the ongoing search progress too much, the ratio of the number of the immigrants to the population size, i.e., the replacement rate, is usually set to a small value, e.g., 0.2 [30,32] or 0.3 [31]. However, the most appropriate replacement rate may vary at different stages of evolutionary process. For example, the replacement rate should be small when there is no change, while a large replacement rate would be more preferable when a change occurs, so that the population will have higher probability to move to new promising area. Hence, adapting the replacement rate during the evolutionary process might be a better choice that deserves some further investigation.

For the third concern, there are two questions: which individuals should be replaced by the newly introduced immigrants and when should the replacement take place. Regarding the first question, two commonly used methods are to replace random individuals and the worst individuals of the population, respectively. The former approach aims at increasing the diversity of the population whereas the search progress might be disturbed. For example, in the stationary environment (i.e., between two consecutive changes), fitter individuals should be kept in order to pass down their useful genes to the next generation. However, to replace random individuals in the population with immigrants may drive away these fitter individuals, and thus disturbs the current search progress. The latter approach tries to minimize the interruption of the search progress and meanwhile increase the diversity of the population via immigrants. Furthermore, in [33], the individuals are numbered and the worst individual and its next neighbors are replaced by immigrants, leading the proposed GA to present a kind of self-organized behavior called self-organized criticality (SOC), which helps to maintain the diversity of the population in dynamic environments. With regard to when to employ replacement, immigrants are usually created at each generation and replacement takes place after that [8,30–32]. In this way, the diversity of the population can be maintained throughout the whole search progress.

The survival probability of the immigrants after being introduced into the current population is another important issue, yet easily tends to be neglected. For example, when the population locates in the area near the local optimum, the best fitness found by the population might be much higher than the mean fitness of all possible solutions of the search space. Therefore, the probability that new immigrants have higher fitness than individuals in the current population is generally small. However, these immigrants may take potentially useful ingredients with them though they might have lower fitness, and thus, they should be protected from being eliminated via selection. Taking binary deceptive function for example, some low-fitness immigrants may take schemata which help to form global optimum, so these immigrants should be reserved. In [33], the immigrants are put in a subpopulation and the replacement is avoided for individuals between subpopulation and the main population. In this way, new immigrants are protected from being replaced by fitter individuals.

## 3 Measures for understanding the behavior of EAs in dynamic environments

In order to analyze and compare the performance of EAs in dynamic fitness landscapes, appropriate measures must be decided first, such as "collective mean fitness" [34], "optimization accuracy, stability and reactivity" [35], and "accuracy and adaptability" [36]. More details about the overview of existing measures for EAs in dynamic environments can be found in [34,35].

Generally speaking, researchers view the behavior of EAs from two different perspectives [37]. Some pay more attention on extreme behaviors of the system, in particular, the best that system can do. These measures are preferred by application practitioners who wonder the best results their systems can obtain. Others are concerned for measures which can characterize the population as a whole, e.g., average, standard deviations, and distributions. These measures are often adopted when EAs are understood as representations of evolutionary systems. In this case, EAs are used to model these systems and hence the behavior of the best individual is not so important. On the basis of these considerations, measures in this paper except *diversity*, i.e., *performance* and *robustness*, are all examined in the way of both a best measure and an average measure. These measures are all based upon discussions in [37] and are formulated as follows.

First, performance is the standard measure of how well the system can do. Simply and intuitively, fitness function is used as a measure of performance in this paper. The overall *Best Performance* and the overall *Average Performance* of an algorithm on a DOP are defined as

$$\overline{F}_{\mathrm{BOG}} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N}F_{\mathrm{BOG}_{ij}}\right) \tag{1}$$

and

$$\overline{F}_{\mathrm{Avg}} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N}F_{\mathrm{Avg}_{ij}}\right) \tag{2}$$

respectively, where $G$ is the total number of generations for a run, $N$ is the total number of runs, and $F_{\mathrm{BOG}_{ij}}$ and $F_{\mathrm{Avg}_{ij}}$ are the best-of-generation fitness and the average fitness of the population of generation $i$ of run $j$, respectively.

Second, robustness is to some extent complicated, since it has many different notions and hence its definition depends on a particular problem. Jen [38] stated that "robustness is an approach to feature persistence in systems that compels us to focus on perturbations, and assemblages of perturbations, to the system different from those considered in the design of the system, or from those encountered in its prior history". Hence it is necessary to specify both the feature and the perturbation of interest before discussing robustness. In the field of dynamic optimization, an important feature is the performance of the system, which is defined as the fitness in this paper. Moreover, the perturbations in dynamic optimization include system influence and control influence [39], of which the former is considered in this paper. System influence is the response of the dynamic system to the changes over time of itself, and the control influence is the response of the dynamic system at current time to the decisions made by the solver in the past. Based on these considerations, the *Best Robustness* of generation $i$ and the *Average Robustness* of generation $i$ are defined as

$$\overline{R}_{\text{Best}_i} = \frac{1}{N}\sum_{j=1}^{N} R_{\text{Best}_{ij}} \tag{3}$$

and

$$\overline{R}_{\text{Avg}_i} = \frac{1}{N}\sum_{j=1}^{N} R_{\text{Avg}_{ij}} \tag{4}$$

respectively, where $R_{\text{Best}_{ij}}$ and $R_{\text{Avg}_{ij}}$ are the *Best Robustness* and *Average Robustness* of generation $i$ of run $j$, respectively, which are defined as

$$R_{Best_{ij}} = \begin{cases} 1, & \text{if } \frac{F_{\text{BOG}_{ij}}}{F_{\text{BOG}_{i-1j}}} > 1 \\ \frac{F_{\text{BOG}_{ij}}}{F_{\text{BOG}_{i-1j}}}, & \text{otherwise} \end{cases} \tag{5}$$

and

$$R_{Avg_{ij}} = \begin{cases} 1, & \text{if } \frac{F_{\text{Avg}_{ij}}}{F_{\text{Avg}_{i-1j}}} > 1 \\ \frac{F_{\text{Avg}_{ij}}}{F_{\text{Avg}_{i-1j}}}, & \text{otherwise} \end{cases}, \tag{6}$$

respectively. From the equations defining robustness above, we can see that higher robustness level indicates more persistent fitness level.

Finally, diversity is a measure of how different of individuals of the population are. It indicates how much of the search space the EA is now exploring. In this paper, the diversity of generation $i$ is defined as

$$\overline{\text{Div}}_i = \frac{1}{N}\sum_{j=1}^{N} \text{Div}_{ij} \tag{7}$$

where $\text{Div}_{ij}$ is the diversity of generation $i$ of run $j$, and for binary encoding, $\text{Div}_{ij}$ can be calculated as

$$\text{Div}_{ij} = \frac{1}{ln(n-1)}\sum_{p=1}^{n}\sum_{q\neq p}^{n} \text{HD}(p,q) \tag{8}$$

where $l$ is the encoding length, $n$ is the population size, and $\text{HD}(p,q)$ is the Hamming distance between the $p$th and $q$th individuals in the population.

**Table 1** Pseudo-code for SGA

```
begin
   t := 0 and initialize population P(0) randomly
   evaluate population P(0)
   repeat
      P'(t) := selectForReproduction(P(t))
      crossover(P'(t),p^c)      // p^c is the crossover probability
      mutate(P'(t),p^m)       // p^m is the mutation probability
      evaluate population P'(t)
      P(t+1) := P'(t)
   until the termination condition is met      // e.g., t > t_max
end
```

## 4 Description of algorithms investigated

All algorithms to be investigated derive from the standard genetic algorithm (SGA). SGA progresses via selecting and recombining a population of candidate solutions. The population is initialized randomly. In each generation, parents are selected based on their fitness, and based on these parents, offsprings are created via crossover and mutation. This procedure is iteratively repeated until a certain stop criterion is satisfied, e.g., the max predefined number of generations $t_{\text{max}}$ is reached. Table 1 shows the pseudo-code for SGA, where $p^c$ and $p^m$ are the probability of crossover and mutation, respectively.

### 4.1 Genetic algorithms with direct immigrants scheme

As discussed in Sect. 2, the *direct immigrants scheme* generates immigrants based on the current population. The simplest way is to use individuals in the current population as the base to generate immigrants. In this paper, the GA with elitism-based immigrants (denoted EIGA) in [30] and the GA with individual information-based hybrid immigrants (denoted IIHIGA) in [31] are re-investigated. EIGA aims at improving the performance of GAs in slightly or slowly changing environments while IIHIGA tries to make GAs more adapted in more severely changing environments. The pseudo-code of them is shown in Table 2.

Within EIGA, for each generation $t$, after normal genetic operations, the elite $E(t-1)$ from previous generation is used as the base to create immigrants. By mutating $E(t-1)$ bitwise with a probability $p^m_{ei}$, a set of $r_{ei} \times n$ individuals are iteratively generated, where $n$ is the population size and $r_{ei}$ is the replacement rate. Then the worst individuals in the current population are replaced with these newly introduced immigrants. On the other hand, within IIHIGA, for each generation $t$, apart from elitism-based immigrants generated in the same way as in EIGA, $r_{ri} \times n$ random immigrants and $r_{di} \times n$ dualism-based immigrants are also generated, where $n$ is the population size and $r_{ri}$ and $r_{di}$ are the ratios of the number of random immigrants and dualism-based immigrants to the population size respectively. Dualism-based immigrants

**Table 2** Pseudo-code for GAs with elitism-based immigrants (EIGA) and hybrid individual information-based immigrants (IIHIGA)

```
begin
  t := 0 and initialize population P(0) randomly
  evaluate population P(0)
  repeat
    P'(t) := selectForReproduction(P(t))
    crossover(P'(t),p^c)      // p^c is the crossover probability
    mutate(P'(t),p^m)      // p^m is the mutation probability
    evaluate population P'(t)

    // generate elitism-based immigrants
    denote the elite in P(t − 1) by E(t − 1)
    generate r_ei × n immigrants by mutating E(t − 1) with p^m_ei
    evaluate these elitism-based immigrants

    if the hybrid scheme is used then      // for IIHIGA
      generate r_ri × n random immigrants
      evaluate these random immigrants
      generate r_di × n immigrants by mutating the dual of
        E(t − 1) with p^m_di
      evaluate these dualism-based immigrants
    end if

    replace the worst individuals in P'(t) with the generated
      immigrants
    P(t + 1) := P'(t)
  until the termination condition is met      // e.g., t > t_max
end
```

**Table 3** Pseudo-code for GAs with environmental information-based immigrants (EIIGA) and hybrid environmental information-based immigrants (EIHIGA)

```
begin
  t := 0 and initialize population P(0) randomly
  evaluate population P(0)
  repeat
    P'(t) := selectForReproduction(P(t))
    crossover(P'(t),p^c)      // p^c is the crossover probability
    mutate(P'(t),p^m)      // p^m is the mutation probability
    evaluate population P'(t)

    extract the allele distribution vector D_P(t) from P'(t)
    generate r_eii × n immigrants by sampling the allele
      distribution vector D_P(t)
    evaluate these environmental information-based immigrants

    if complementary environmental information used then
      // for EIHIGA
      generate r_ceii × n immigrants by sampling the
        complementation of D_P(t)
      evaluate these complementary environmental
        information-based immigrants
    end if

    replace the worst individuals in P'(t) with the generated
      immigrants
    P(t + 1) := P'(t)
  until the termination condition is met      // e.g., t > t_max
end
```

are generated from mutating the dual of the elite $E(t-1)$ bitwise with a probability $p^m_{di}$. The dual of an individual is the one that is symmetric to it with respect to the central point of the search space. Specifically, given a binary-encoded individual $\boldsymbol{x} = (x_1, \ldots, x_l) \in I = \{0, 1\}^l$ of length $l$, its dual $\boldsymbol{x}^d$ is defined as

$$\boldsymbol{x}^d = \text{dual}(\boldsymbol{x}) = (x^d_1, \ldots, x^d_l) \in I, \tag{9}$$

where $x^d_i = 1 - x_i (i = 1, \ldots, l)$. Then these three types of immigrants are used to replace the worst individuals in the current population. In IIHIGA, the replacement rate $r_i = r_{ei} + r_{ri} + r_{di}$, and the ratios of the number of three kinds of immigrants to the population size are adaptively adjusted based on their relative performance within the range of $[r_{\min}, r_i - 2r_{\min}]$, where $r_{\min}$ is the minimum ratio of the number of one type of immigrants to the population size. If the three immigrants schemes tie, no changes of ratios occur, otherwise, for the worst two immigrants schemes, $r_{xi}$ (i.e., $r_{ri}$, $r_{ei}$, or $r_{di}$) will be reduced by $r_{xi} - \max\{r_{\min}, r_{xi} - \alpha\}$, where $\alpha$ is a constant value, and the ratio for the winner immigrants scheme will be increased to make the replacement rate $r_i$ fixed.

## 4.2 Genetic algorithms with indirect immigrants scheme

As examples of the *indirect immigrants scheme*, the GA with environmental information-based immigrants (EIIGA) and the GA with environmental information-based hybrid immigrants (EIHIGA) in [32] are investigated in this paper. The goal of EIIGA is to enhance the performance of GAs in slowly or slightly changing environments while EIHIGA attempts to improve the performance of GAs in more severely changing environments. The pseudo-code of them is shown in Table 3.

Within EIIGA, the allele distribution in the population is calculated at first and then acts as the base to generate immigrants. For generation $t$, after normal genetic operations, the allele distribution vector $\boldsymbol{D}_P(t)$ is extracted from current population and for binary encoding, the frequency of ones over the population in a gene locus can be regarded as the allele distribution for that locus. Then a set of $r_{eii} \times n$ individuals are generated by sampling $\boldsymbol{D}_P(t)$, where $n$ is the population size and $r_{eii}$ is the replacement rate. A new individual $S = \{s_1, \ldots, s_l\}$ is created from $\boldsymbol{D}_P(t) = \{d^P_1, \ldots, d^P_l\}$ ($l$ is the encoding length) as follows:

$$s_i = \begin{cases} 1, & \text{if rand}[0.0, 1.0] < d^P_i \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The generated individuals then act as immigrants and replace the worst individuals in the current population.

Within EIHIGA, in addition to $r_{eii} \times n$ immigrants created via sampling the allele distribution vector $\boldsymbol{D}_P(t)$ of the population, $r_{ceii} \times n$ immigrants are also created via sampling the complementary allele distribution vector $\boldsymbol{D}^c_P(t)$ of $\boldsymbol{D}_P(t)$ in the same way as shown in equation (10), where $r_{ceii}$ is the ratio of the number of complementary environmental information-based immigrants to the population size and $\boldsymbol{D}^c_P(t) = \boldsymbol{1} - \boldsymbol{D}_P(t)$. These two sets of immigrants will then replace the worst individuals in the current population. In EIHIGA, the replacement rate $r_i = r_{eii} + r_{ceii}$, and $r_{eii}$ and $r_{ceii}$ are adaptively adjusted based on the performance of corresponding immigrants schemes within the range $[r_{\min}, r_i - r_{\min}]$, where $r_{\min}$ is the minimum ratio of the number of immigrants of one type to the population size. If one immigrants scheme performs worse than the other, its relative ratio $r_{xi}$ (i.e., $r_{eii}$ or $r_{ceii}$) will be reduced by $r_{xi} - \max\{r_{\min}, r_{xi} - \alpha\}$, where $\alpha$ is a constant value, meanwhile the winner immigrants scheme will increase its ratio accordingly to ensure that $r_i$ is fixed. If the two immigrants schemes tie, no changes to ratios occur. As another example of the *indirect immigrants scheme*, memory-based immigrants scheme for GAs [24] mainly aims at addressing DOPs in cyclic environments and is not in the scope of study of this paper.

### 4.3 Genetic algorithms with both direct and indirect immigrants schemes

In the field of automation, robotics, mechanics, and manufacturing, many researchers have been engaging in studying the tradeoff of performance and robustness over the years [40–44]. Under definitions in this paper, we believe the performance and the robustness are still to some extent conflicting with each other. Specifically, algorithms reveal better performance on DOPs lose more robustness. Intuitively speaking, EAs with direct immigrants schemes might beat those with indirect immigrants schemes with respect to the performance while be beaten by them with respect to the robustness.

On the other hand, direct immigrants schemes investigated in this paper can be regarded as using the individual information to generate immigrants, and indirect immigrants schemes investigated in this paper can be viewed as utilizing the environmental information (i.e., the allele distribution is treated as the representation of the environment) to create immigrants. Since the evolution of a population can be recognized as the process of the interaction between the environment and individuals, we propose a new immigrants scheme for EAs in dynamic environments, which hybrids immigrants generated based on two kinds of immigrants schemes.

Taking the interaction between individuals and the environment into account, we expect our new approach will show much better performance, or will at least strike a balance between the performance and the robustness. The proposed immigrants scheme just simply hybridizes the elitism-based, the dualism-based, the environmental information-based and the complementary environmental information-based immigrants in each generation. This GA with hybrid immigrants is denoted as HIGA in this paper and its pseudo-code is shown in Table 4.

Within HIGA, for each generation $t$, after normal genetic operations, $r_{ei} \times n$ elitism-based immigrants, $r_{di} \times n$ dualism-based immigrants, $r_{eii} \times n$ environmental information-based immigrants, and $r_{ceii} \times n$ complementary information-based immigrants are generated, where $n$ is the population size and the replacement rate $r_i = r_{ei} + r_{di} + r_{eii} + r_{ceii}$. Then the worst individuals of the population are replaced with these immigrants. Similar to other immigrants schemes investigated in this paper, the ratios of the number of four kinds of immigrants to the population size are adaptively adjusted based on their relative performance within the range of $[r_{\min}, r_i - 3r_{\min}]$, where $r_{\min}$ is the minimum ratio of the number of one type of immigrants to the population size. If four immigrants schemes tie, no changes of ratios occur, otherwise, for the worst three immigrants schemes, $r_{xi}$ (i.e., $r_{ei}$, $r_{di}$, $r_{eii}$ or $r_{ceii}$) will be reduced by $r_{xi} - \max\{r_{\min}, r_{xi} - \alpha\}$, where $\alpha$ is a constant value, and the ratio for the winner immigrants scheme will be increased to make the replacement rate $r_i$ fixed. All in all, algorithms investigated in this paper are summarized in Table 5.

**Table 4** Pseudo-code for GA with hybrid immigrants (HIGA)

```
begin
  t := 0 and initialize population P(0) randomly
  evaluate population P(0)
  repeat
    P'(t) := selectForReproduction(P(t))
    crossover(P'(t),p^c)      // p^c is the crossover probability
    mutate(P'(t),p^m)         // p^m is the mutation probability
    evaluate population P'(t)

    denote the elite in P(t-1) by E(t-1)
     generate r_ei × n immigrants by mutating E(t-1) with p^m_ei
     generate r_di × n immigrants by mutating the dual of
         E(t-1) with p^m_di
     extract the allele distribution vector D_P(t) from P'(t)
     generate r_eii × n immigrants by sampling the allele
         distribution vector D_P(t)
     generate r_ceii × n immigrants by sampling the
         complementation of D_P(t)
     evaluate all the generated immigrants

     replace the worst individuals in P'(t) with the generated
      immigrants
     P(t+1) := P'(t)
  until the termination condition is met      // e.g., t > t_max
end
```

**Table 5** Summary of immigrants schemes investigated

| Classification | Denotement and corresponding full name |
|---|---|
| Direct | EIGA [30] (elitism-based immigrants) |
| | IIHIGA [31] (individual information-based hybrid immigrants) |
| Indirect | EIIGA [32] (environmental information-based immigrants) |
| | EIHIGA [32] (environmental information-based hybrid immigrants) |
| Hybrid | HIGA (hybrid immigrants scheme) |

## 5 Dynamic test environments

### 5.1 Dynamic environments generator

Over the ages, in addition to developing approaches into EAs to deal with DOPs, many researchers have been applying their energies to conceiving dynamic test environments to study the performance of the developed approaches. Generally speaking, there are three types of generators for dynamic test environments.

For the first type of dynamic environment generators, the environment just switches between two or more states of a problem. For example, many researchers have frequently used dynamic knapsack problem as the test environment, where the weight capacity of the knapsack oscillates between two or more fixed values [10,15,16]. For this type of generators, environmental dynamics are characterized by the speed of change measured in EA generations.

The second type of generators construct dynamic environments based on a predefined fitness landscape. The base landscape is usually defined in $n$-dimensional real space and is made up of a number of component landscapes, each of which can change its own morphology independently with such parameters as peak height, peak slope, and peak location. The optimum solution of the landscape is the center of the peak with the highest height [13,45,46]. For this type of generators, environmental dynamics are characterized by the step size of parameter change and the speed of changes in EA time.

The third type of generator was proposed in [17] and [47], which can construct dynamic environments from any binary-encoded stationary function $f(\boldsymbol{x})$ $(\boldsymbol{x} \in \{0,1\}^l)$ by a bitwise exclusive-or (XOR) operator. Suppose the environment changes periodically every $\tau$ generations, the dynamics can be formulated as follows:

$$f(\boldsymbol{x}, t) = f(\boldsymbol{x} \oplus \boldsymbol{M}(k)), \tag{11}$$

where $\oplus$ is the XOR operator (i.e., $1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 0 = 0$), $k = \lceil t/\tau \rceil$ is the index of the period, $t$ is the current count of generations, and $\boldsymbol{M}(k)$ is the XORing mask for period $k$. An XORing mask $\boldsymbol{M}(k)$ can be generated incrementally as follows:

$$\boldsymbol{M}(k) = \boldsymbol{M}(k-1) \oplus \boldsymbol{T}(k), \tag{12}$$

where $\boldsymbol{T}(k)$ is an intermediate binary template randomly created for period k containing $\rho \times l$ ones. At first environmental period, i.e., $k = 1$, $\boldsymbol{M}(1)$ is initialized to be a zero vector indicating there is no change in the environment.

With this XOR DOP generator, the speed of the environmental changes is controlled by the parameter $\tau$ while the severity of the environmental changes is controlled by the parameter $\rho \in [0.0, 1.0]$. Bigger $\rho$ indicates severer changes while smaller $\tau$ means faster changes. It can be seen that the XOR DOP generator does not change the search space. It just rotates the candidate solutions by some degree before each function evaluation. For example, if $l = 100$, $\tau = 10$ and $\rho = 0.5$, then after every 10 generations, values of 50 randomly chosen bits of each individual will be flipped (i.e., $1 \rightarrow 0$ and $0 \rightarrow 1$).

### 5.2 Generating dynamic test environments for experiments

#### *5.2.1 The OneMax function*

The OneMax function is a function for a binary string $\boldsymbol{x}$ of length $L$. The goal is to maximize the number of ones in a string. In this paper, we assume $L = 100$ and thus, the function is defined as follows:

$$\text{maximize } f(\boldsymbol{x}) = \sum_{i=1}^{100} x_i, \tag{13}$$

where $f(\boldsymbol{x})$ is the fitness of a binary string $\boldsymbol{x} = (x_1, \ldots, x_{100}) \in I = \{0,1\}^{100}$.

#### *5.2.2 The Royal Road function*

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 [48]. 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 $c_i = 4$ $(i = 1, \ldots, 25)$ to the total fitness if and only if every bit is one. Therefore, the fitness of a string $\boldsymbol{x}$ is the sum of the coefficients $c_i$ corresponding to each given schema $s_i$, of which $\boldsymbol{x} \in s_i$, i.e.:

$$\text{maximize } f(\boldsymbol{x}) = \sum_{i=1}^{25} c_i \delta_i(\boldsymbol{x}), \tag{14}$$

where

$$\delta_i(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in s_i \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

#### *5.2.3 The deceptive function*

Deceptive functions are a family of GA-hard functions where there exists low-order schemata that instead of combining to form high-order schemata, form schemata resulting in a solution called deceptive attractor [49], which is sub-optimal itself or near a sub-optimal solution. A 4-bit fully deceptive function can be defined as follows:

$$f(\boldsymbol{x}) = \begin{cases} 4, & \text{if } u(\boldsymbol{x}) = 4 \\ 3 - u(\boldsymbol{x}), & \text{otherwise.} \end{cases} \tag{16}$$

where $u(\boldsymbol{x})$ is the unitation function, which returns the number of ones in the string $\boldsymbol{x}$. In this paper, a deceptive function is constructed consisting of 25 copies of the above 4-bit fully deceptive function. The fitness is the sum of contribute of each sub-problem. The maximum fitness is 100 for the Deceptive function as well as for the OneMax and Royal Road function in this paper.

#### *5.2.4 Constructing dynamic test environments*

Since the three stationary problems above are all binary functions, the XOR DOP generator is used in this paper. Generally speaking, the difficulty of the three stationary problems above for EAs is increasing in the order from OneMax to Royal Road to Deceptive. The fitness landscape of each stationary problem periodically changes every $\tau$ generations during the run. To study every algorithm's capability of adapting to dynamic environment at different searching stages, $\tau$ is set to 10 and 50 respectively. To test each algorithm's performance under different degree of changes, $\rho$ is set to 0.1, 0.2, 0.5, and 1.0 respectively for each run of an algorithm on a problem, representing different degree of changes from slight ($\rho = 0.1$) to medium ($\rho = 0.2, 0.5$) to significant ($\rho = 1.0$). In total, we systematically generate a series of 8 DOPs, 2 values of $\tau$ with 4 values of $\rho$, from each stationary test problem.

## 6 Experimental study on algorithms investigated

Since the behaviors of EIGA, EIIGA, IIHIGA and EIHIGA in dynamic environments have already been solely and thoroughly studied [30–32], we mainly focus on comparing the behaviors between two types of immigrants schemes and studying the effect of the interaction between them.

### 6.1 Experimental design

In the experiments, all algorithms were investigated on DOPs constructed above. The parameters' settings for all algorithms are shown in Table 6. Note that population size $n = 100$ and replacement rate $r_i = 0.3$ hold for all algorithms such that each algorithm has 130 evaluations per generation.

For each algorithm on a DOP, 30 independent runs were executed with the same set of random seeds. For each run of an algorithm on a DOP, 50 environmental changes were allowed and in order to calculate experimental results according to measures of performance aforementioned, the best-of-generation fitness, the average fitness of the population, and the diversity of the population were recorded every generation.

### 6.2 Experimental analysis regarding comparisons between two types of immigrants schemes

The experimental results with respect to overall performance are presented in Table 7. The best results among algorithms in each environment are shown in bold. The statistical results of comparing algorithms with respect to performance via non-parametric Wilcoxon rank sum tests at a 0.05 level of significance are given in Table 8. In Table 8, the result regarding algorithm 1 - algorithm 2 is marked as "$s+$" or "$s-$" when algorithm 1 is significantly better than or significantly worse than algorithm 2, respectively, and "$\sim$" indicates there is no statistical difference between two algorithms. To better understand the behaviors of algorithms, the dynamic *Best Performance* regarding the best-of-generation fitness against generations and *Average Performance* regarding the average fitness of the population against generations of algorithms on DOPs for the first 10 environments with $\tau = 50$, $\rho = 0.1$ and $\rho = 1$ are plotted in Figs. 2 and 3, respectively, where the data were averaged over 30 runs. The dynamic population diversity of algorithms against generations on DOPs for the first 10 environments with $\tau = 50$ and $\rho = 0.5$ is plotted in Fig. 1, where the data were averaged over 30 runs. From Tables 7 and 8 and Figs. 1, 2, and 3, we can get some observations by comparing the performance between two types of immigrants schemes.

First, regarding the *Average Performance*, IIHIGA outperforms EIHIGA and EIGA outperforms EIIGA on almost all DOPs. On the other hand, regarding the *Best Performance*, IIHIGA outperforms EIHIGA and EIGA outperforms EIIGA on all dynamic Deceptive problems and most dynamic OneMax problems, see statistical-test results regarding EIGA-EIIGA and IIHIGA-EIHIGA in Table 8. This is because immigrants generated via direct immigrants schemes are more concentrated in the new promising searching space than those generated via indirect immigrants schemes. In some cases, direct immigrants schemes can even generate the optimal individual for a new environment. Therefore, direct immigrants schemes can quickly focus the searching force of EAs on the new optimal area and move the population there. This can be obtained from Figs. 2 and 3 that after a change of the environment, algorithms with direct immigrants schemes recover more quickly and can reach a higher fitness level than algorithms with indirect immigrants schemes. This result can be also observed from the dynamic population diversity in Fig. 1 that after a change of the environment, the diversity of the population using indirect immigrants schemes is increasing for a moment at first, implying the new promising area is been searching, and then decreases, indicating some fitter individuals have already been found. On the other hand, after a change of the environment, the diversity of the population using direct immigrants schemes promptly arrives at a high level and then decreases immediately, which indicates that

**Table 6** Parameters for all algorithms investigated

| Common settings | Algorithms | Other settings |
|---|---|---|
| Generational | EIGA | $r_{ei} = r_i = 0.3$, $p^m_{ei} = 0.01$ |
| Uniform crossover, $p^c = 0.6$ | EIIGA | $r_{eii} = r_i = 0.3$ |
| Bit flip mutation, $p^m = 0.01$ | EIHIGA | $r_{eii} = r_{ceii} = 0.15$ initially |
| | | $[r_{\min}, r_i - r_{\min}] = [0.04, 0.26]$, $\alpha = 0.02$ |
| Fitness proportional selection implemented via stochastic universal sampling algorithm with elitism of size 1 | IIHIGA | $r_{ri} = r_{ei} = r_{di} = 0.1$ initially |
| | | $[r_{\min}, r_i - 2r_{\min}] = [0.04, 0.22]$, $\alpha = 0.02$ |
| | | $p^m_{ei} = p^m_{di} = 0.01$ |
| Chromosome length, $l = 100$ | HIGA | $r_{eii} = r_{ceii} = r_{ei} = r_{di} = 0.075$ initially |
| Population size, $n = 100$ | | $[r_{\min}, r_i - 3r_{\min}] = [0.04, 0.18]$, $\alpha = 0.02$ |
| Replacement rate, $r_i = 0.3$ | | $p^m_{ei} = p^m_{di} = 0.01$ |

**Table 7** Experimental results with respect to overall performance

| GAs and functions | OneMax | | | | Royal Road | | | | Deceptive | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Overall *Best Performance* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | **90.2** | **79.4** | 64.5 | 55.4 | **56.8** | 40.9 | 26.5 | 45.8 | **68.8** | **60.2** | 53.8 | 86.2 |
| EIIGA | 85.5 | 77.7 | 67.7 | 60.8 | 55.3 | **41.6** | 29.0 | 45.5 | 60.9 | 55.4 | 51.4 | 80.6 |
| EIHIGA | 85.1 | 77.2 | **68.3** | 68.2 | 56.0 | 41.4 | **30.1** | 76.7 | 56.6 | 52.7 | 50.7 | 57.6 |
| IIHIGA | 88.4 | 77.8 | 67.9 | **92.2** | 54.2 | 39.0 | 29.1 | 82.6 | 68.2 | 58.6 | **54.4** | 88.4 |
| HIGA | 88.3 | 77.8 | 68.0 | 90.0 | 54.2 | 39.5 | 28.9 | **83.1** | 67.5 | 58.2 | 54.2 | **88.5** |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | **98.4** | **96.0** | 85.6 | 65.8 | **87.8** | **73.9** | 50.2 | 43.6 | 76.1 | 74.6 | 72.7 | 87.4 |
| EIIGA | 95.4 | 92.1 | 82.1 | 67.9 | 83.5 | 72.5 | 51.9 | 45.7 | 71.6 | 68.7 | 64.2 | 84.1 |
| EIHIGA | 95.3 | 91.8 | 82.8 | 95.9 | 83.7 | 72.7 | **54.0** | 94.6 | 65.6 | 63.3 | 58.1 | 62.1 |
| IIHIGA | 98.0 | 95.3 | **87.2** | **98.4** | 87.0 | 73.6 | 53.7 | 96.6 | **84.5** | **80.1** | **75.4** | **91.1** |
| HIGA | 98.0 | 95.2 | 86.9 | 98.0 | 87.3 | 73.8 | 53.8 | **96.8** | 83.9 | 79.3 | 74.4 | 90.7 |
| Overall *Average Performance* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | **87.0** | **75.7** | **60.2** | 50.8 | **49.6** | **33.9** | **20.4** | 40.8 | **64.3** | **54.4** | **47.2** | **82.6** |
| EIIGA | 78.5 | 69.7 | 58.7 | 51.0 | 39.4 | 27.0 | 16.2 | 34.0 | 52.3 | 44.5 | 38.8 | 68.9 |
| EIHIGA | 75.8 | 67.4 | 57.9 | 53.3 | 38.8 | 25.7 | 15.8 | 52.9 | 45.0 | 40.3 | 37.4 | 42.6 |
| IIHIGA | 83.3 | 72.0 | 60.0 | 80.8 | 44.1 | 29.4 | 19.2 | 72.6 | 57.9 | 48.7 | 43.5 | 70.6 |
| HIGA | 83.1 | 71.5 | 59.4 | **83.6** | 43.5 | 29.2 | 18.5 | **73.1** | 56.0 | 47.4 | 42.5 | 67.7 |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | **96.5** | **93.8** | **82.5** | 62.0 | **81.6** | **67.9** | 45.4 | 40.0 | **73.8** | **71.7** | **68.7** | **84.1** |
| EIIGA | 90.1 | 86.2 | 74.8 | 59.2 | 67.6 | 56.8 | 38.4 | 35.2 | 65.6 | 61.4 | 53.8 | 74.1 |
| EIHIGA | 86.9 | 83.0 | 73.2 | 84.0 | 65.5 | 54.8 | 38.3 | 75.4 | 56.2 | 53.0 | 45.6 | 46.7 |
| IIHIGA | 94.3 | 91.2 | 82.0 | 93.6 | 77.2 | 64.2 | **45.6** | **87.2** | 72.9 | 68.7 | 64.2 | 78.9 |
| HIGA | 94.7 | 91.3 | 81.6 | **94.6** | 76.8 | 63.9 | 45.0 | 86.9 | 70.1 | 66.0 | 61.6 | 75.7 |

Bold values are significant at $\alpha = 0.05$ by Wilcoxon rank sum test

some fitter individuals or even the new optimal individuals are already in the population. Also, it can be clearly seen that algorithms with indirect immigrants schemes maintain higher diversity level than algorithms with direct immigrants schemes. This interesting result indicates that approaches that try to maintain higher diversity level in the population do not naturally lead to better performance of EAs in dynamic environments.

**Table 8** The statistical tests of comparing the performance of algorithms on DOPs

| GAs and functions | OneMax | | | | Royal Road | | | | Deceptive | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Best Performance* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s+$ | $s+$ | $s-$ | $s-$ | $s+$ | $s-$ | $s-$ | $\sim$ | $s+$ | $s+$ | $s+$ | $s+$ |
| IIHIGA–EIHIGA | $s+$ | $s+$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIGA | $s-$ | $s-$ | $s+$ | $s+$ | $s-$ | $s-$ | $s+$ | $s+$ | $s-$ | $s-$ | $s+$ | $s+$ |
| HIGA–EIIGA | $s+$ | $\sim$ | $s+$ | $s+$ | $s-$ | $s-$ | $\sim$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIHIGA | $s+$ | $s+$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–IIHIGA | $\sim$ | $\sim$ | $s+$ | $s-$ | $\sim$ | $s+$ | $\sim$ | $s+$ | $s-$ | $s-$ | $\sim$ | $\sim$ |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s+$ | $s+$ | $s+$ | $s-$ | $s+$ | $s+$ | $s-$ | $s-$ | $s+$ | $s+$ | $s+$ | $s+$ |
| IIHIGA–EIHIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s-$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIGA | $s-$ | $s-$ | $s+$ | $s+$ | $s-$ | $\sim$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIHIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $\sim$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–IIHIGA | $\sim$ | $s-$ | $s-$ | $s-$ | $s+$ | $s+$ | $\sim$ | $s+$ | $s-$ | $s-$ | $s-$ | $\sim$ |
| *Average Performance* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s+$ | $s+$ | $s+$ | $s-$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| IIHIGA–EIHIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIGA | $s-$ | $s-$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s-$ |
| HIGA–EIIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s-$ |
| HIGA–EIHIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–IIHIGA | $\sim$ | $s-$ | $s-$ | $s+$ | $s-$ | $\sim$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s-$ |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| IIHIGA–EIHIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIGA | $s-$ | $s-$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s-$ |
| HIGA–EIIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIHIGA | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ | $s+$ |
| HIGA–IIHIGA | $s+$ | $s+$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ |

Second, indirect immigrants schemes investigated in this paper seem not suitable for dynamic Deceptive problems. This can be clearly observed from Table 7 that the difference of the overall performance is very obvious between algorithms with direct immigrants schemes and the corresponding algorithms with indirect immigrants schemes. At the same time, seeing dynamic performance of algorithms on Deceptive problems in Figs. 2 and 3, algorithms with indirect immigrants schemes keep much lower fitness level than the corresponding algorithms with direct immigrants schemes. Worse off, when $\rho = 1$ EIHIGA even can not obtain a satisfying result. This is because immigrants based on environmental information and its complementation can easily break global optimal building blocks existing in the Deceptive functions via crossover.

Third, on dynamic Royal Road problems with respect to the *Best Performance*, the situation seems six of one, half a dozen of the other for algorithms with direct immigrants schemes and indirect immigrants schemes. Specifically, algorithms with direct immigrants schemes outperform algorithms with indirect immigrants schemes in most cases when $\tau = 50$ while are beaten mostly when $\tau = 10$, see statistical-test results of the *Best Performance* on dynamic Royal Road problems regarding EIGA–EIIGA and IIHIGA–EIHIGA in Table 8. Taking into account the landscape features of the Royal Road problem and the working mechanism of the two types of immigrants schemes, these results can be explained as follows. The Royal Road problem in this paper features schema hierarchies and intermediate stepping stones. It can be represented as "a tree of increasingly higher-

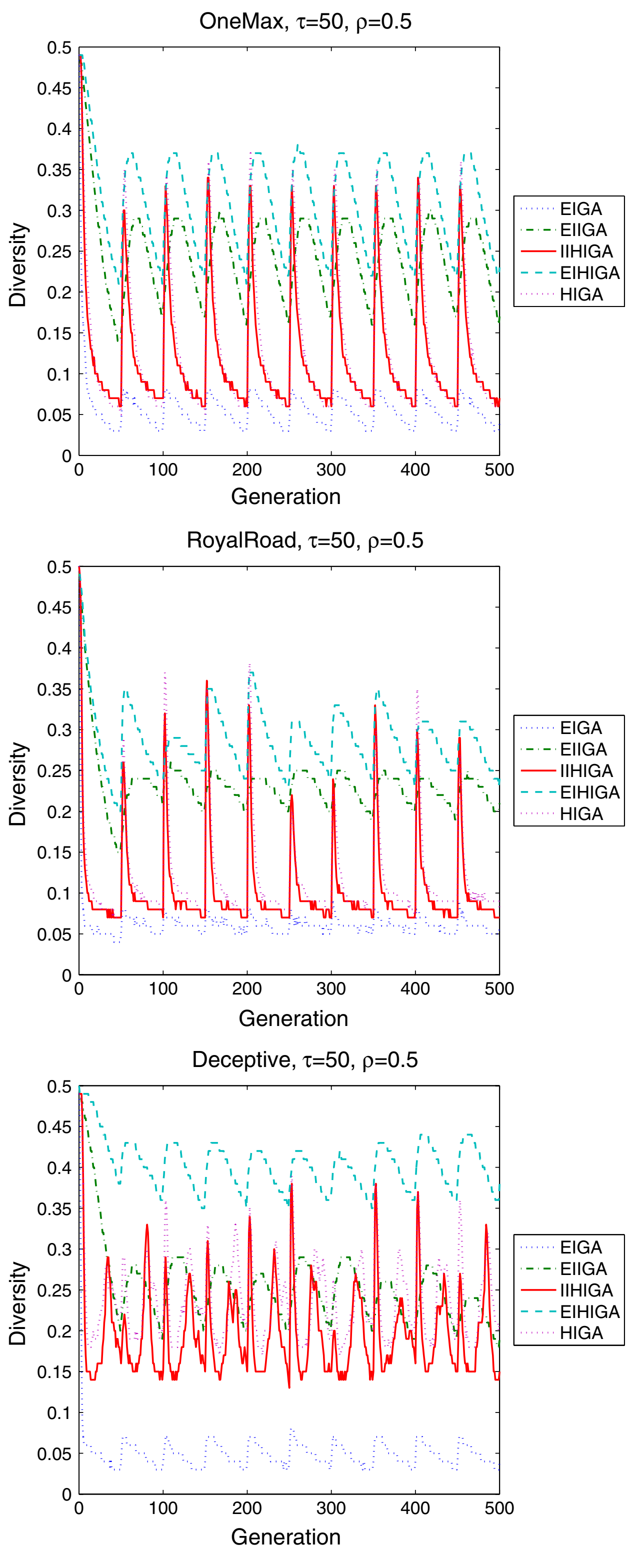


**Fig. 1** Dynamic population diversity of algorithms on DOPs with $\tau = 50$ and $\rho = 0.5$ for the first ten environments

order schemas, with schemas of each order being composable to produce schemas of the next higher order" [48]. In other words, there are many plateaus between fit schemata. To jump out of these plateaus will cost substantial time via bitwise mutation (which is adopted by direct immigrants schemes in this paper, i.e., immigrants are generated by bit flip mutation of some individuals) with probability $\frac{1}{l}$, where $l$ is the chromosome length [50,51]. Nevertheless, the capability to maintain higher diversity level of indirect immigrants schemes in this paper can help the population to jump out of such plateaus. At the early searching stage, the population is dominated by lower-order schemata, and hence, jumping out of plateaus is more likely to produce higher-order schemata. As the search goes on, higher-order schemata become dominating the population. Therefore, jumping out of plateaus can sometimes yield lower-order schemata. This is why when $\tau = 10$ algorithms with indirect immigrants schemes perform better while when $\tau = 50$ they are surpassed by the corresponding algorithms with direct immigrants schemes in performance. This can be also observed from Fig. 2 that on the Royal Road problem algorithms with indirect immigrants schemes keep higher fitness level at first and lower fitness later on than the corresponding algorithms with direct immigrants schemes.

To compare the robustness of the two types of immigrants schemes, the experimental results with respect to overall robustness are presented in Table 9. The best results among algorithms in each environment are shown in bold. The statistical results of comparing algorithms with respect to robustness via nonparametric Wilcoxon rank sum tests at a 0.05 level of significance are given in Table 10. In Table 10, the result regarding algorithm 1–algorithm 2 is marked as "$s+$" or "$s-$" when algorithm 1 is significantly better than or significantly worse than algorithm 2 regarding *Best Robustness* and *Average Robustness*, respectively, and "$\sim$" indicates there is no statistical difference between two algorithms. Note that only the robustness at generations when there are changes are used for calculation since the robustness in this paper is mainly a measure of how the system responds to changes of the environment. The dynamic *Best Robustness* regarding the *Best Robustness* against generations and the dynamic *Average Robustness* regarding the *Average Robustness* against generations on DOPs for all the environments with $\tau = 50$, $\rho = 0.1$ and $\rho = 1$ are plotted in Figs. 4 and 5 respectively, where the data were averaged over 30 runs and only the robustness at generations when there are changes is plotted for the same reason above.

Comparing the robustness of the two types of immigrants schemes, EAs with indirect immigrants schemes outperforms those with corresponding direct immigrants schemes on nearly all DOPs with respect to both *Best Robustness* and *Average Robustness*, see statistical-test results regarding EIGA–EIIGA and IIHIGA–EIHIGA in Table 10. Seeing Figs. 2 and 3, when the environment changes, except for the dynamic Deceptive problem with $\tau = 50$ and $\rho = 1$, it is clear that the performance of direct immigrants schemes

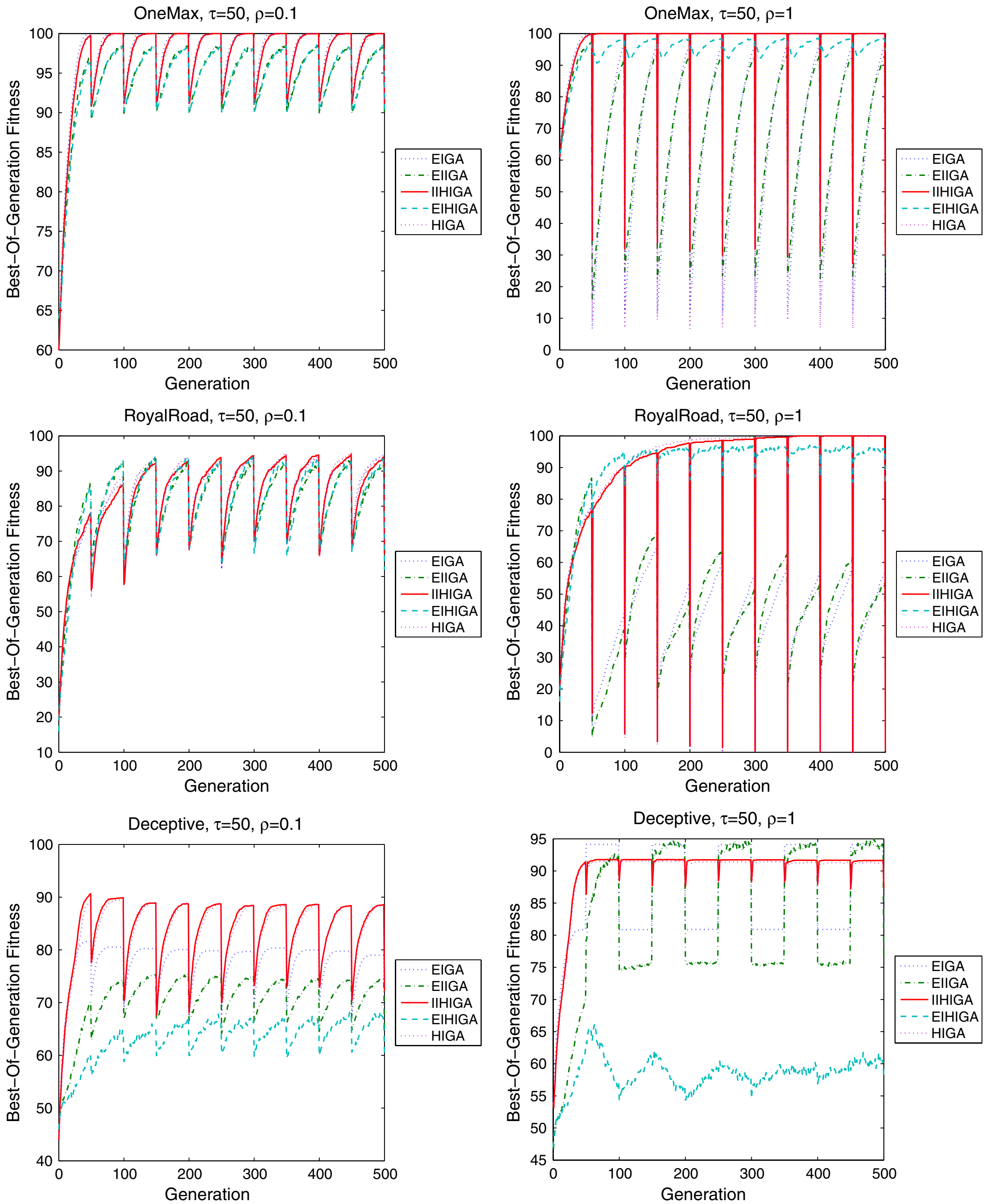


**Fig. 2** Dynamic *Best Performance* of algorithms on DOPs with $\tau = 50$ and $\rho = 0.1$ and 1, respectively for the first ten environments

drops more sharply than the corresponding indirect immigrants schemes. This result can also be apparently observed from dynamic robustness in Figs. 4 and 5 that indirect immigrants schemes manage to keep higher robustness level than the corresponding direct immigrants schemes except for the dynamic Deceptive problem with $\tau = 50$ and $\rho = 1$. These phenomena can be regarded as the result of different capability of immigrants schemes to maintain the diversity of the population which has already been examined above. To sum up, the results above confirm our expectation in Sect. 4 that the performance and the robustness can not be optimized simultaneously. Generally speaking, EAs with direct immigrants schemes have higher performance level and lower robustness level than EAs with indirect immigrants schemes.

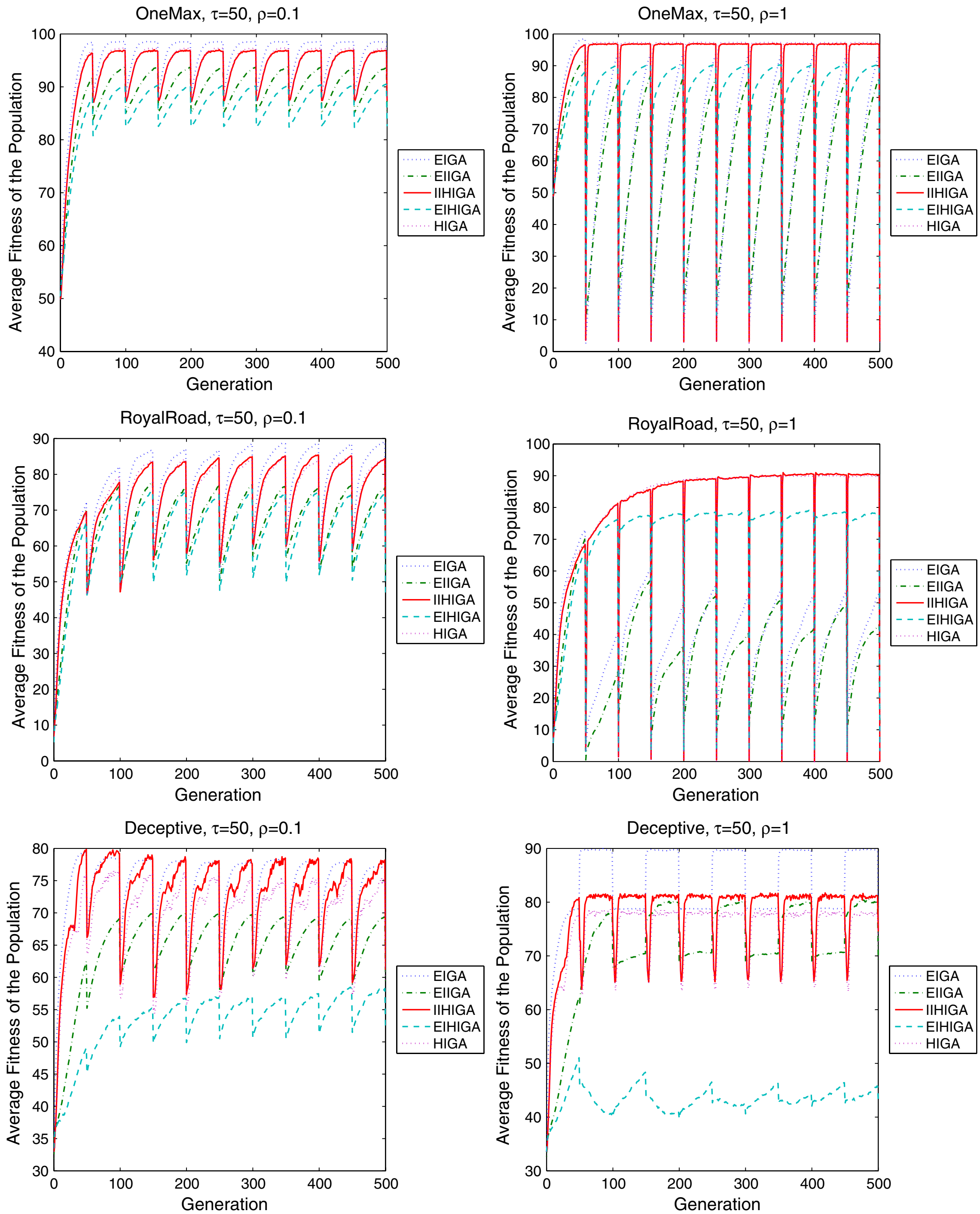


**Fig. 3** Dynamic *Average Performance* of algorithms on DOPs with $\tau = 50$ and $\rho = 0.1$ and 1, respectively for the first ten environments

### 6.3 Experimental analysis regarding the interaction between two types of immigrants schemes

In order to investigate the interaction between two types of immigrants schemes, the dynamic *Best Performance* and *Average Performance* of immigrants in investigated algorithms on DOPs for the first 10 environments with $\tau = 50$, $\rho = 0.1$ and $\rho = 1$ are plotted in Figs. 6 and 7, where "HIGADirect" and "HIGAIndirect" denote immigrants in HIGA generated directly and indirectly respectively, and the data were averaged over 30 runs.

First, comparing the proposed algorithm with indirect immigrants schemes, HIGA outperforms EIIGA and EIHIGA on nearly all DOPs with respect to the *Average*

**Table 9** Experimental results with respect to overall robustness

| GAs and functions | OneMax | | | | Royal Road | | | | Deceptive | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Overall *Best Robustness* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | 0.92 | 0.87 | 0.78 | 0.71 | 0.73 | 0.57 | 0.40 | 0.81 | 0.87 | 0.75 | 0.67 | 0.96 |
| EIIGA | **0.94** | **0.90** | **0.84** | 0.78 | **0.79** | **0.68** | 0.52 | 0.83 | 0.92 | 0.88 | 0.86 | 0.95 |
| EIHIGA | **0.94** | **0.90** | **0.84** | **0.93** | **0.79** | 0.67 | **0.53** | **0.87** | **0.94** | **0.92** | **0.91** | 0.96 |
| IIHIGA | 0.93 | 0.87 | 0.77 | 0.35 | 0.74 | 0.58 | 0.38 | 0.07 | 0.84 | 0.76 | 0.69 | 0.96 |
| HIGA | 0.93 | 0.88 | 0.77 | 0.12 | 0.74 | 0.59 | 0.39 | 0.07 | 0.84 | 0.77 | 0.71 | **0.97** |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | 0.91 | 0.82 | 0.55 | 0.14 | 0.71 | 0.48 | 0.20 | 0.38 | 0.87 | 0.73 | 0.50 | 0.93 |
| EIIGA | **0.92** | **0.84** | **0.61** | 0.26 | **0.72** | 0.52 | **0.25** | 0.41 | 0.89 | 0.78 | 0.60 | 0.91 |
| EIHIGA | **0.92** | **0.84** | **0.61** | **0.97** | **0.72** | **0.53** | **0.25** | **0.90** | **0.90** | **0.82** | **0.72** | 0.95 |
| IIHIGA | 0.91 | 0.82 | 0.56 | 0.33 | 0.70 | 0.48 | 0.19 | 0.03 | 0.80 | 0.64 | 0.49 | 0.96 |
| HIGA | 0.91 | 0.82 | 0.55 | 0.10 | 0.70 | 0.48 | 0.20 | 0.02 | 0.80 | 0.64 | 0.50 | **0.97** |
| Overall *Average Robustness* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | 0.92 | 0.86 | 0.75 | 0.66 | 0.69 | 0.49 | 0.25 | 0.70 | 0.84 | 0.71 | 0.59 | **0.96** |
| EIIGA | 0.93 | 0.88 | 0.80 | **0.73** | **0.74** | **0.58** | 0.33 | **0.73** | 0.91 | 0.85 | 0.82 | **0.96** |
| EIHIGA | **0.94** | **0.89** | **0.81** | 0.68 | **0.74** | 0.56 | **0.34** | 0.08 | **0.93** | **0.91** | **0.89** | **0.96** |
| IIHIGA | 0.92 | 0.87 | 0.73 | 0.07 | 0.69 | 0.49 | 0.22 | 0.04 | 0.82 | 0.73 | 0.62 | 0.93 |
| HIGA | 0.92 | 0.87 | 0.73 | 0.06 | 0.69 | 0.50 | 0.22 | 0.03 | 0.83 | 0.74 | 0.64 | 0.94 |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA | 0.90 | 0.81 | 0.52 | 0.09 | 0.66 | 0.42 | 0.12 | **0.26** | 0.84 | 0.69 | 0.43 | **0.94** |
| EIIGA | 0.91 | 0.82 | 0.57 | **0.19** | **0.68** | 0.45 | **0.14** | 0.23 | 0.87 | 0.74 | 0.52 | 0.93 |
| EIHIGA | **0.92** | **0.83** | **0.59** | 0.15 | **0.68** | **0.47** | **0.14** | 0.06 | **0.89** | **0.79** | **0.65** | 0.93 |
| IIHIGA | 0.91 | 0.81 | 0.53 | 0.05 | 0.66 | 0.41 | 0.11 | 0.02 | 0.76 | 0.59 | 0.43 | 0.93 |
| HIGA | 0.90 | 0.81 | 0.53 | 0.05 | 0.66 | 0.42 | 0.11 | 0.02 | 0.77 | 0.61 | 0.45 | **0.94** |

Bold values are significant at $\alpha = 0.05$ by Wilcoxon rank sum test

*Performance* and on most DOPs with respect to the *Best Performance*. On the other hand, compared with direct immigrants schemes, HIGA outperforms EIGA on most DOPs with respect to the *Best Performance* while is beaten on nearly all DOPs with respect to the *Average Performance*. Meanwhile, HIGA is beaten by IIHIGA on almost all DOPs except dynamic Royal Road problems regarding the *Best Performance* and on nearly all DOPs with respect to the *Average Performance*, see statistical-test results regarding HIGA–EIGA, HIGA–EIIGA, HIGA–EIHIGA and HIGA–IIHIGA in Table 8. However, seeing the dynamic *Best Performance* and *Average Performance* in Figs. 2 and 3, things are not so simple as what results in Table 8 show. HIGA maintains nearly the same dynamic *Best Performance* level as IIHIGA and sometimes even higher *Best Performance* level than IIHIGA on dynamic OneMax and Royal Road problems when there is no change. Even if on dynamic Deceptive problems, IIHIGA keeps only a little higher *Best Performance* level than HIGA when there is no change. Therefore, the degradation in overall *Best Performance* of HIGA over IIHIGA is due to its larger fitness drop than IIHIGA when change occurs, which can be clearly observed in Fig. 2, especially when $\rho = 1$. This can be also seen from the dynamic *Best Performance* of immigrants in Fig. 6 that on DOPs when $\tau = 50$ and $\rho = 1$ "HIGADirect" maintains higher fitness level than IIHIGA or nearly the same fitness level as IIHIGA in stationary environmental period while drops more largely than IIHIGA. On the other hand, on most DOPs IIHIGA has much higher *Average Performance* level than HIGA. Nevertheless, seeing the dynamic *Average Performance* of immigrants in Fig. 7, "HIGADirect" maintains much higher level than IIHIGA. Hence the degradation in *Average Performance* of HIGA over IIHIGA is the result of much lower *Average Performance* level of immigrants generated indirectly, see the curve "HIGAIndirect" in Fig. 7.

Second, examining the interaction between direct immigrants schemes and indirect immigrants schemes, in most cases, immigrants generated directly in HIGA performs better than those in IIHIGA with respect to the *Best Performance*, especially with respect to the *Average Performance*.

**Table 10** The statistical tests of comparing the robustness of algorithms on DOPs

| GAs and functions | OneMax | | | | Royal Road | | | | Deceptive | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Best Robustness* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $\sim$ |
| IIHIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $\sim$ |
| HIGA–EIGA | $s+$ | $s+$ | $s-$ | $s-$ | $s+$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ | $s+$ | $s+$ |
| HIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–IIHIGA | $\sim$ | $s+$ | $\sim$ | $s-$ | $\sim$ | $s+$ | $s+$ | $\sim$ | $\sim$ | $s+$ | $s+$ | $s+$ |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| IIHIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–EIGA | $\sim$ | $\sim$ | $\sim$ | $s-$ | $s-$ | $\sim$ | $\sim$ | $s-$ | $s-$ | $s-$ | $\sim$ | $s+$ |
| HIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–IIHIGA | $\sim$ | $\sim$ | $s-$ | $s-$ | $\sim$ | $\sim$ | $s+$ | $s-$ | $\sim$ | $\sim$ | $s+$ | $s+$ |
| *Average Robustness* | | | | | | | | | | | | |
| $\tau = 10, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $\sim$ |
| IIHIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ |
| HIGA–EIGA | $\sim$ | $s+$ | $s-$ | $s-$ | $\sim$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ | $s+$ | $s-$ |
| HIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ |
| HIGA–IIHIGA | $\sim$ | $\sim$ | $\sim$ | $s-$ | $\sim$ | $s+$ | $\sim$ | $s-$ | $s+$ | $s+$ | $s+$ | $s+$ |
| $\tau = 50, \rho \Rightarrow$ | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 | 0.1 | 0.2 | 0.5 | 1 |
| EIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ | $s-$ | $s-$ | $s-$ | $s+$ |
| IIHIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $\sim$ |
| HIGA–EIGA | $\sim$ | $\sim$ | $s+$ | $s-$ | $\sim$ | $\sim$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ | $\sim$ |
| HIGA–EIIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–EIHIGA | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s-$ | $s+$ |
| HIGA–IIHIGA | $s-$ | $\sim$ | $\sim$ | $\sim$ | $\sim$ | $s+$ | $\sim$ | $\sim$ | $s+$ | $s+$ | $s+$ | $+$ |

On the other hand, immigrants generated indirectly in HIGA performs much better than those in EIHIGA on all DOPs with respect to the *Best Performance*. These results can be also observed from Figs. 6 and 7 that "HIGADirect" maintains nearly the same fitness level as IIHIGA or higher fitness level than IIHIGA on most DOPs regarding both kinds of performance, and "HIGAIndirect" has much higher fitness level than EIHIGA regarding the *Best Performance*. These phenomena demonstrate the positive effect on the performance of algorithms via the interaction between the *Direct Immigrants Scheme* and the *Indirect Immigrants Scheme*.

More concretely, the interaction between two types of immigrants schemes probably lies on the way in which the population distribution vector $\boldsymbol{D}_P$ is working. Instead of incrementally updated, $\boldsymbol{D}_P$ in this paper is updated globally. Specifically, in each generation, a new vector $\boldsymbol{D}_P$ is constructed based on the whole new population. Consequently, immigrants generated directly are involved when constructing $\boldsymbol{D}_P$. As aforementioned, immigrants generated directly are generally fit in the new environment, and thus individuals generated via sampling this vector are naturally fitter than immigrants in EIHIGA. On the other hand, individuals generated via sampling $\boldsymbol{D}_P$ are more close to a point in the searching space whereas individuals generated via mutating a individual will sometimes produce individuals far apart from promising area. This will lead to a greedy behavior of the algorithm, which is beneficial to the local performance.

Finally, comparing the robustness of HIGA with other two types of immigrants schemes, HIGA performs worse than EIHIGA and EIIGA, and better than IIHIGA and EIGA on most DOPs with respect to both *Best Robustness* and *Average Robustness*, see statistical-test results regarding HIGA–EIHIGA, HIGA–EIIGA, HIGA–IIHIGA and HIGA–EIGA

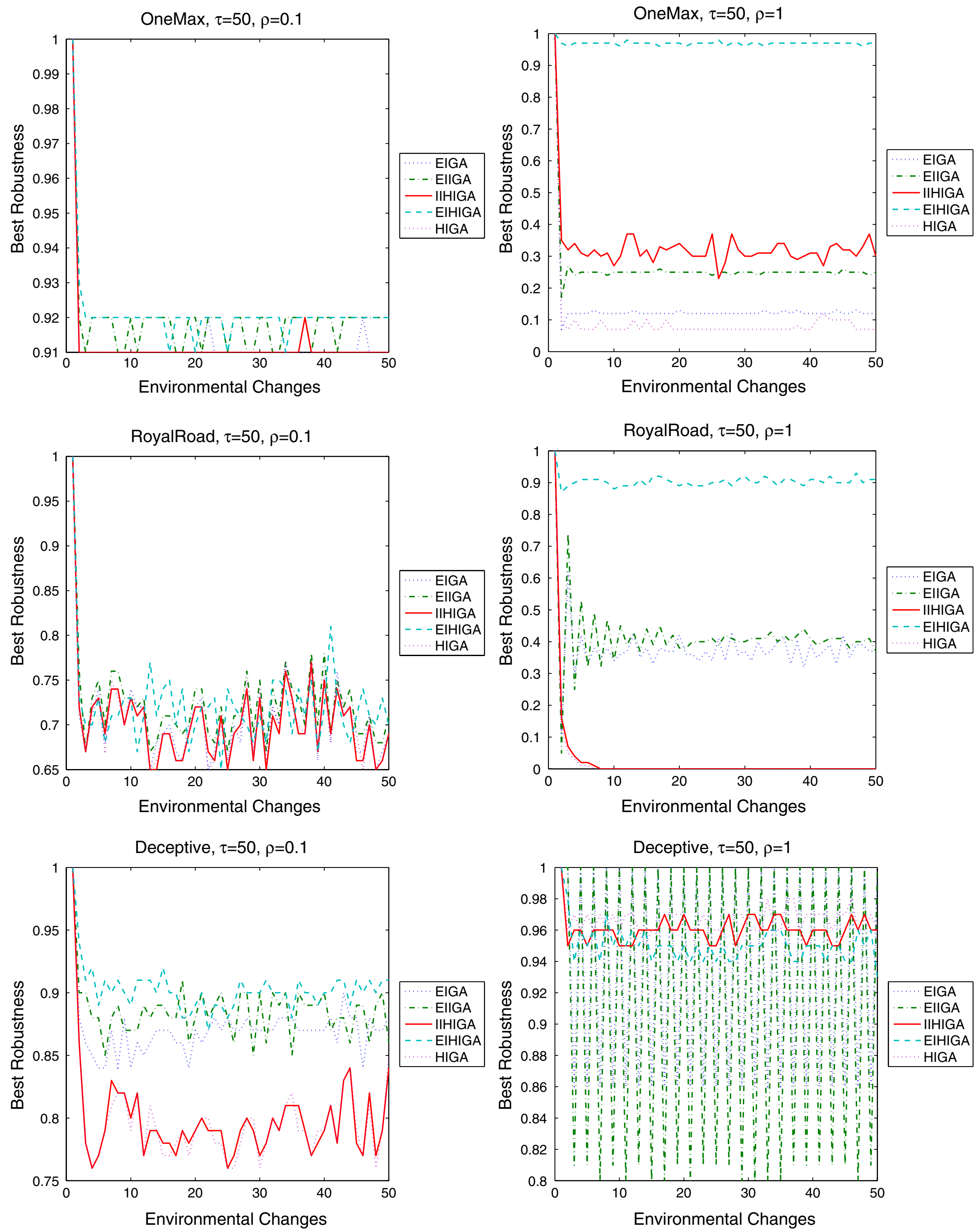


**Fig. 4** Dynamic *Best Robustness* of algorithms on DOPs with $\tau = 50$ and $\rho = 0.1$ and 1, respectively

in Table 10. These results can be explained from the dynamic diversity of the population in Fig. 1. We can see that HIGA has much lower diversity level than EIHIGA and EIIGA, and it maintains a little higher diversity level than IIHIGA and much higher level than EIGA. In a word, the results of HIGA confirm our expectation in Sect. 4 that taking into account the interaction between two types of immigrants can benefit the performance of the algorithm in dynamic environments. However, to pay a price for this promotion in performance, HIGA can not maintain satisfying robustness level as EAs with indirect immigrants schemes. More interestingly, in some cases, HIGA can outperform EA with

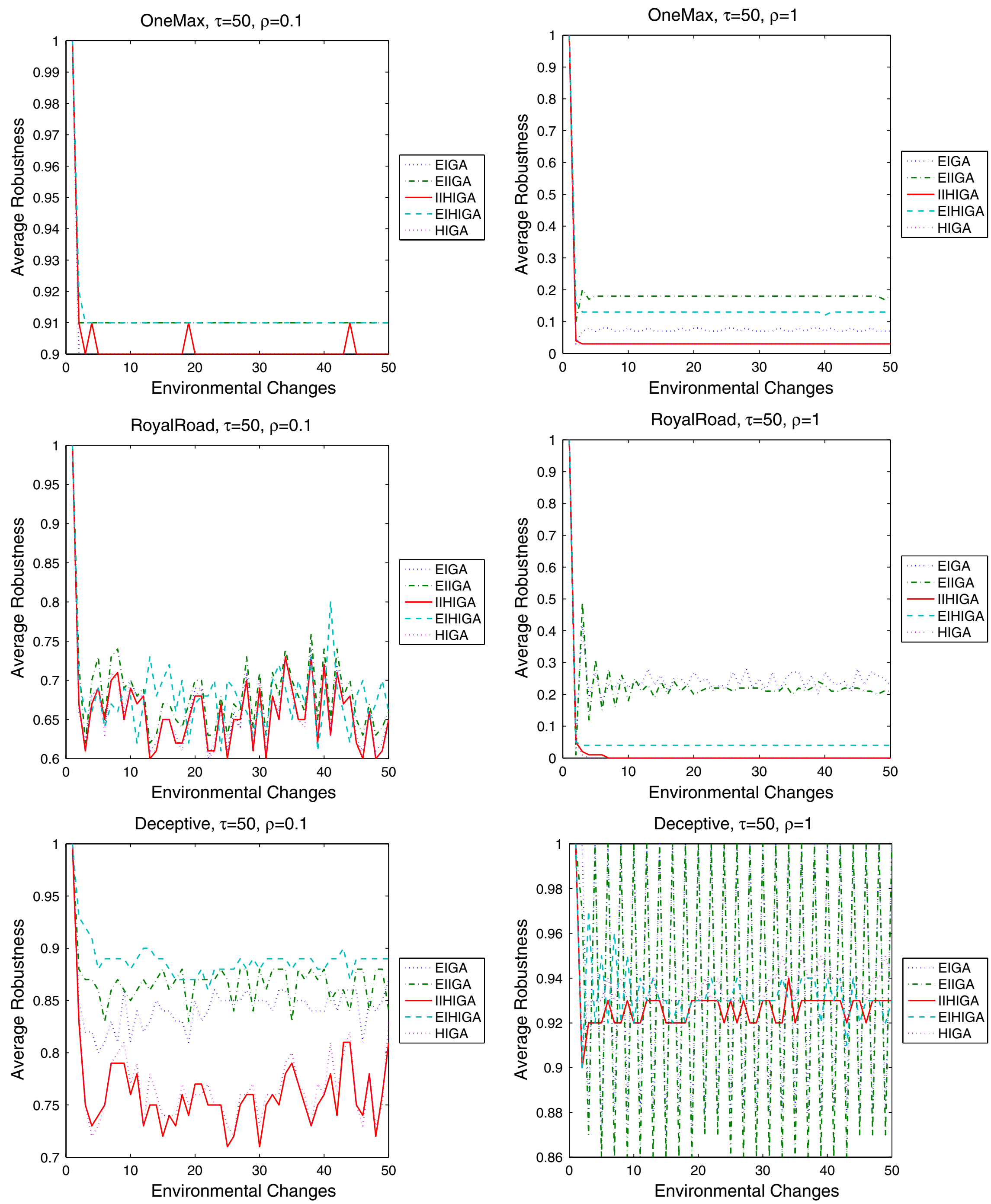


**Fig. 5** Dynamic *Average Robustness* of algorithms on DOPs with $\tau = 50$ and $\rho = 0.1$ and 1, respectively

direct immigrants scheme regarding both the performance and the robustness.

### 6.4 Experimental analysis regarding common behaviors of algorithms in dynamic environments

First, comparing the extreme behavior with the average behavior of algorithms on DOPs, it can be seen from Figs. 2 and 3 that the *Average Performance* of an algorithm on a DOP falls greater than the *Best Performance* of it on the same DOP when any changes occur. This result can be more clearly observed via comparing Fig. 4 with Fig. 5. On most DOPs and for most algorithms, the *Best Robustness* keeps at a higher level than the corresponding *Average Robustness*. This is because when the environment changes, most individuals which are adapted to the old new environment will

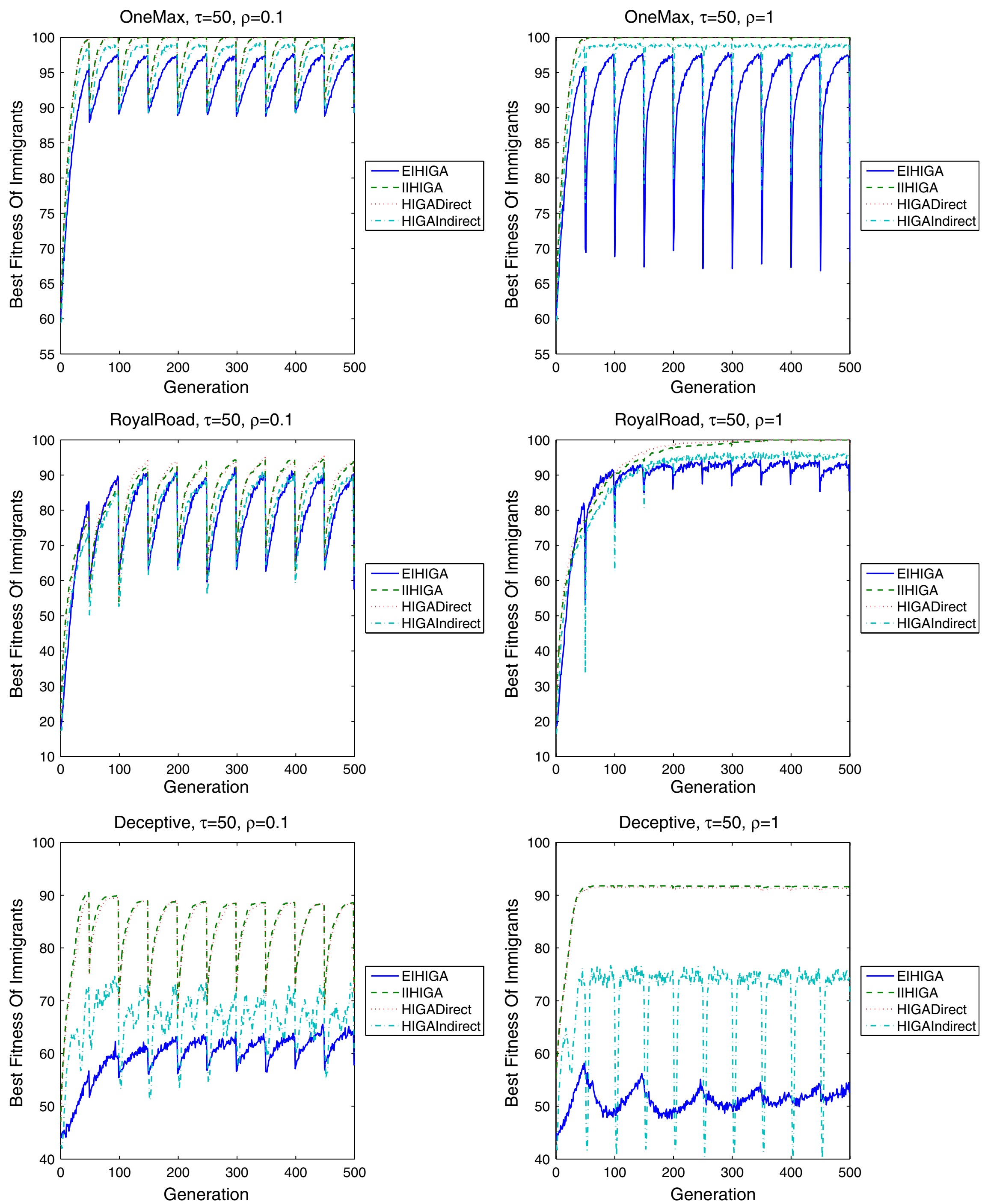


**Fig. 6** Dynamic *Best Performance* of immigrants in HIGA, EIHIGA, IIHIGA on DOPs ($\tau = 50$, $\rho = 0.1$ and 1) for the first ten environment

lose their adaption to the new environment and thus their fitness falls greatly. However, there are some individuals which will be more fit in the new environment than they are in the old environment, these potential new best individuals will mitigate the fall of the best performance.

Second, taking stock of dynamic *Best Robustness* and *Average Robustness* of algorithms on DOPs, in most cases, decreases of robustness are relatively small at first, then become larger, and finally small again. The probable reason can be depicted as follows. The XOR operator used in experiments influences a individual via changing intermediate schema in it. At early searching stage, the population is dominated by individuals that have a few elementary schemata and very few intermediate schemata, and then the

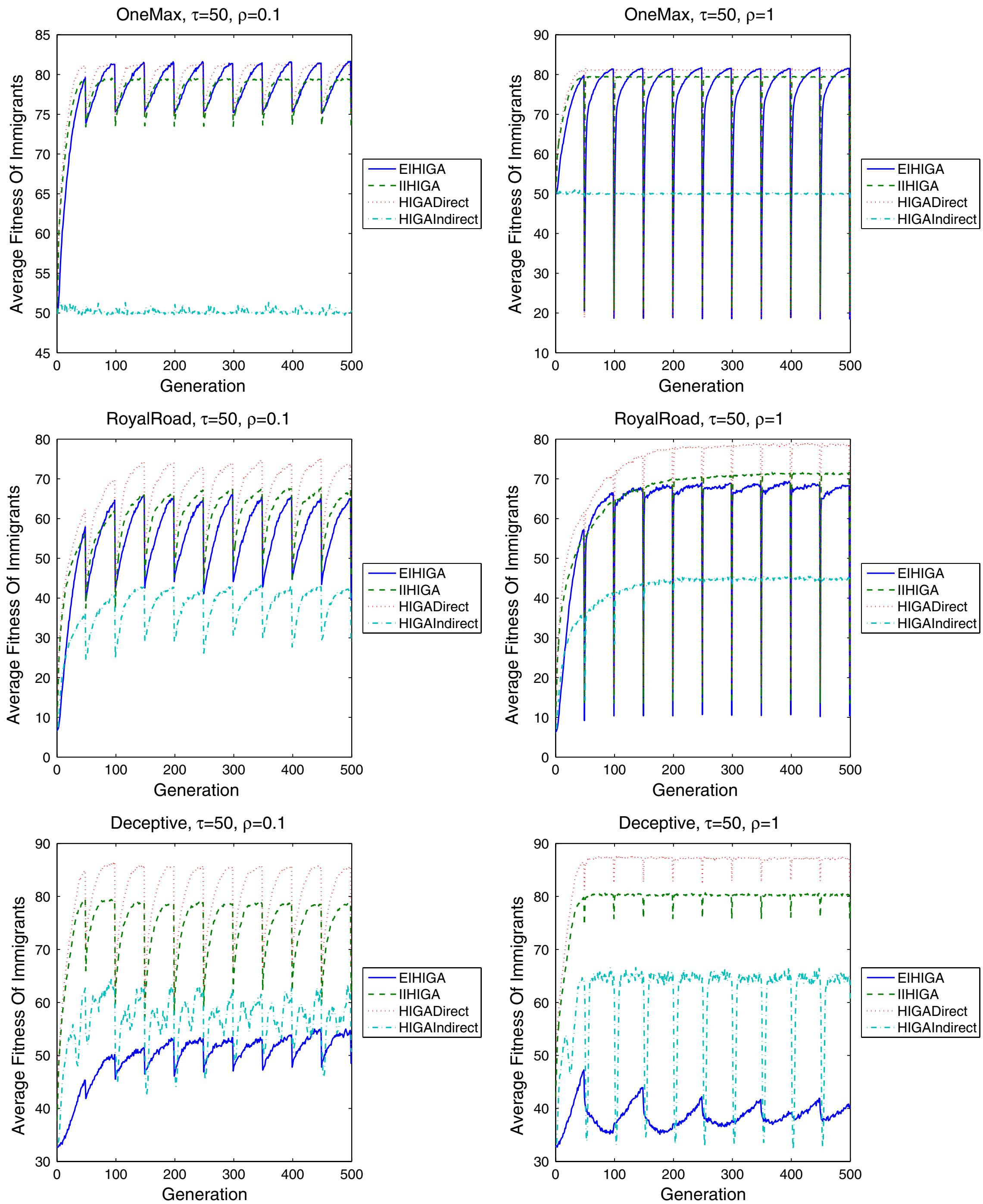


**Fig. 7** Dynamic *Average Performance* of immigrants in HIGA, EIHIGA, IIHIGA on DOPs ($\tau = 50$, $\rho = 0.1$ and 1) for the first ten environment

algorithm is devoting itself to exploring particular intermediate schemata. Therefore, the robustness decreases slightly first and then significantly. However, as the searching process goes on, the algorithm has found most of the intermediate schemata. In this case, imposing XOR operator on these intermediate schemata would most likely to result in intermediate schemata which already exist in the population, and hence the robustness decreases slightly.

## 6.5 Experimental analysis regarding the general effect of environmental dynamics on the performance of algorithms

Carefully studying results in Table 7, and examining the effect of dynamic environments on the performance of investigated algorithms, several results can be observed as follows. First, for a DOP with fixed $\rho$, the performance of algorithms rises as $\tau$ increases from 10 to 50. This result is natural, since when the environment changes slowly, i.e., $\tau = 50$, the algorithm has enough time to attain higher fitness level before any changes occur. Second, for a DOP with fixed $\tau$, the performance of algorithms generally decreases as $\rho$ increases from 0.1 to 0.2 to 0.5. It is easy to understand, since a bigger value of $\rho$ means more severely changing environment. However, when $\rho = 1$, the performance of algorithms seems much better, especially on dynamic Royal Road and Deceptive problems. This indicates that environmental changes especially significant changes sometimes can help the population to jump out of particular schemata and to explore a wider range of beneficial intermediate schemata. This can be also observed from dynamic performance in Figs. 2 and 3 that on dynamic Deceptive problem with $\tau = 50$ and $\rho = 1$, the performance of EIGA and EIIGA arises abruptly after the first change of the environment and some environmental changes thereafter. Besides, on dynamic Royal Road problem with $\tau = 50$ and $\rho = 1$, after some environmental changes, the performance of EIGA and EIIGA does not fall or only drops slightly, but then the performance of them will attain a much higher level until the next change of the environment.

Finally, with the same $\rho$ and $\tau$, the performance of algorithms on dynamic OneMax problems is usually better than that of corresponding algorithms on dynamic Royal Road problems. This seems natural, since the stationary OneMax problem is easier for EAs to solve than the stationary Royal Road problem, and for each environmental period every problem can be regarded as a stationary problem. However, on some dynamic Deceptive problems, the performance of algorithms seems better than that on dynamic Royal Road problems. This is because although deceptive attractors are not global optimum, they are suboptimum with relatively higher fitness.

## 7 Conclusions

In this paper, the mechanism of generating immigrants, which is the most important issue among strategies designing immigrants schemes for EAs on DOPs, are closely examined. According to the way in which immigrants are generated, we categorize existing immigrants schemes for EAs on DOPs into direct and indirect schemes, and we investigate them with respect to average behavior and extreme behavior through the experiments. Furthermore, we propose a new immigrants scheme which takes into account the interactions between two types of immigrants schemes for EAs on DOPs. From experimental results, we can draw several conclusions as follows:

First, performance and robustness cannot be optimized simultaneously. Generally speaking, EAs with direct immigrants schemes beat EAs with indirect immigrants schemes with respect to the performance while are beaten by them regarding the robustness. In addition, higher diversity level can mitigate the fall of the robustness of EAs when any changes occur and does not always lead to good performance of EAs in dynamic environments.

Second, the interaction of direct and indirect immigrants schemes does strike a balance between the performance and the robustness. Specifically, it promotes the performance of HIGA over EAs with indirect immigrants schemes at the price of degrading the robustness. On the other hand, this interaction improves the robustness level of HIGA over EAs with direct immigrants schemes while degrades its performance slightly in most cases. More importantly, this interaction reveals positive effect in the performance of EAs, which encourages us to develop more carefully designed interaction between two types of immigrants schemes that will lead to EAs with much better performance in dynamic environments.

Finally, the speed and the severity of the change of the environment, and the difficulty of the base stationary problems can influence the difficulty of DOPs. Generally speaking, fast and severely changing environments pose much of difficulty for EAs. However, in some cases, especially when the environment is changing significantly, better performances of EAs have been observed. This indicates that the change of the environment may sometimes alleviate the difficulty of the problems being solved.

For now, several relevant works remain worthy of future study. A primary work is to explore the potential of interaction of two types of immigrants schemes. Based on the encouraging results presented in this paper, we believe that a more careful design of interaction of two types of immigrants schemes will improve the performance of EAs in dynamic environments. Another future work is to analyze the work theoretically and carry out more comprehensive computational studies using the design of experiment methodology. Besides, finding other forms of indirect immigrants schemes for EAs to address DOPs would be another interesting future work. For example, some prediction model can be used as the base to generate immigrants. Meanwhile, the other three issues of designing immigrants schemes, i.e., the replacement rate, the replacement strategy, and to promote the survival probability of new immigrants, still need to be carefully studied in the future.

Furthermore, since an immigrants scheme mainly concerns incorporating new individuals into the current population, it is naturally applicable for any population-based search algorithm, but not merely EAs. In particular, it would be interesting to apply immigrants schemes to the memetic algorithms (MAs). As a family of meta-heuristic search methods that combine global search strategies (e.g., conventional EAs) with local search heuristics, MAs have been shown to be capable of obtaining high quality solutions more efficiently than conventional EAs [52]. Such characteristic is of great importance in the context of dynamic optimization, for which we are usually provided a tighter time budget than for static optimization. Therefore, it is reasonable to expect that the combination of immigrants schemes and MAs will lead to further improved performance. Two straightforward start points are to make use of a meme to generate new immigrants, and to introduce new memes instead of new individuals to address a dynamic situation.

**Acknowledgments** The authors thank Philipp Rohlfshagen for his insightful comments on an early draft of this paper and are grateful to the anonymous reviewers for providing valuable comments to improve this paper. This work is partially supported by the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033) and an EPSRC grant on "Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications" (EP/E058884/1).

## References

1. Yu JX, Yao X, Choi C-H, Gou G (2003) Materialized view selection as constrained evolutionary optimization. IEEE Trans Syst Man Cybernet Part C 33(4):458–467
2. Tang M, Yao X (2007) A memetic algorithm for VLSI floorplanning. IEEE Trans Syst Man Cybernet Part B 37(1):62–69
3. Jin Y, Branke J (2005) Evolutionary optimization in uncertain environments: a survey. IEEE Trans Evol Comput 9(3):303–317
4. Branke J, Middendorf M, Noeth G, Dessouky M (2005) Waiting strategies for dynamic vehicle routing. Transp Sci 39: 298–312
5. Raman N, Talbot FB (1993) The job shop tardiness problem: a decomposition approach. Eur J Oper Res 69:187–199
6. Cobb HG (1990) An investigation into the use of hypermutation as an adative operator in genetic algorithms having continuous, time-dependent nonstationary environments, Naval Res. Lab., Washington, DC, Tech. Rep. AIC-90-001
7. Vavak F, Fogarty TC, Jukes K (1996) A genetic algorithm with variable range of local search for tracking changing environments. In: Proceedings of the 4th International conference on parallel problem solving from nature, LNCS, vol 1141, pp 376–385
8. Grefenstette JJ (1992) Genetic algorithms for changing environments. Parallel problem solving from nature. Elsevier Science Publishers, The Netherlands, vol 2, pp 137–144
9. Cedeno W, Vemuri VR (1997) On the use of niching for dynamic landscapes. In: Proceedings of the 1997 IEEE international conference on evolutionary computation, pp 361–366
10. Mori N, Kita H, Nishikawa Y (1996) Adaptation to a changing environment by means of the thermodynamical genetic algorithm. Parallel Problem Solving from Nature, ser. LNCS, 1141, pp 513–522
11. Ghosh A, Tsutsui S, Tanaka H (1998) Function optimization in nonstationary environment using steady state genetic algorithms with aging of individuals. In: Proceedings of the 1998 IEEE international conference on evolutionary computation, pp 666–671
12. Ramsey CL, Grefenstette JJ (1993) Case-based initialization of genetic algorithms. In: Proceedings of the 5th international conference on genetic algorithms, Morgan Kaufmann Publishers
13. Branke J (1999) Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 IEEE congress on evolutionary computation, vol 3, pp 1875–1882
14. Bendtsen CN, Krink T (2002) Dynamic memory model for non-stationary optimization. In: Proceedings of the 2002 IEEE congress on evolutionary computation, pp 145–150
15. Goldberg DE, Smith RE (1987) Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Proceedings of the 2nd international conference on genetic algorithms, Lawrence Erlbaum Associates, Mahwah, NJ, USA, pp 59–68
16. Dasgupta D, McGregor D (1992) Nonstationary function optimization using the structured genetic algorithm. In: Proceedings of the 2nd international conference on parallel problem solving from nature, pp 145–154
17. Yang S (2003) Non-stationary problem optimization using the primal-dual genetic algorithm. In: Proceedings of the 2003 IEEE congress on evolutionary computation, vol 3, pp 2246–2253
18. Simões A, Costa E (2007) Improving memory's usage in evolutionary algorithms for changing environments. In: Proceedings of the 2007 IEEE congress on evolutionary computation, pp 276–283
19. Yang S, Yao X (2007) Population-based incremental learning with associative memory for dynamic environments. IEEE Trans Evol Comput (accepted in August 2007)
20. Branke J, Kaußler T, Schmidth C, Schmeck H (2000) A multi-population approach to dynamic optimization problems. In: Proceedings of the adaptive computing in design and manufacturing, pp 299–308
21. Ursem RK (2000) Multinational GAs: multimodal optimization techniques in dynamic environments. In: Proceedings of the 2000 genetic and evolutionary computation conference, pp 19–26
22. Wineberg M, Oppacher F (2000) Enhancing the GA's ability to cope with dynamic environments. In: Proceedings of the 2000 genetic and evolutionary computaion conference, pp 3–10
23. Simõs A, Costa E (2003) An immune system-based genetic algorithm to deal with dynamic environments: diversity and memory. In: Proceedings of the 6th international conference on neural networks and genetic algorithms, pp 168–174
24. Yang S (2005) Memory-based immigrants for genetic algorithms in dynamic environments. In: Proceedings of the 2005 genetic and evolutionary computation conference, vol 2, pp 1115–1122
25. Branke J (2001) Evolutionary optimization in dynamic environments. Kluwer, Norwell, MA
26. Morrison RW (2004) Designing evolutionary algorithms for dynamic environments. Springer, Berlin
27. Weicker K (2003) Evolutionary algorithms and dynamic optimization problems. Der Andere Verlag, Berlin
28. Internet Repository on evolutionary algorithms for dynamic optimization problems (Online). Available: http://www.aifb.uni-karlsruhe.de/~jbr/EvoDOP
29. Cobb HG, Grefenstette JJ (1993) Genetic algorithms for tracking changing environments. In: Proceedings of the 5th international conference on genetic algorithms, pp 523–530
30. Yang S (2007) Genetic algorithms with elitism-based immigrants for changing optimization problems. Applications of Evolutionary Computing, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol 4448, pp 627–636
31. Yang S, Tinós R (2007) A hybrid immigrants scheme for genetic algorithms in dynamic environments. Int J Autom Comput 4(3): 243–254
32. Yu X, Tang K, Yao X (2008) An immigrants scheme based on environmental information for genetic algorithms in changing environments. In: Proceedings of the 2008 IEEE congress on evolutionary computation, pp 1141–1147
33. Tinós R, Yang S (2005) Genetic algorithms with self-organized criticality for dynamic optimization problems. In: Proceedings of the 2005 IEEE congress on evolutionary computation, vol 3, pp 2816–2823

34. Morrison RW (2003) Performance measurement in dynamic environments. In: GECCO workshop on evolutionary algorithms for dynamic optimization problems, pp 5–8
35. Weicker K (2002) Performance measures for dynamic environments. In: Parallel problem solving from nature-PPSN VII, pp 64–73
36. Trojanowski K, Michalewicz Z (1999) Evolutionary algorithms for non-stationary environments. In: Proceedings of 8th workshop: intelligent information systems, pp 229–240
37. Rand W, Riolo R (2005) Measurements for understanding the behavior of the genetic algorithm in dynamic environemnts: a case study using the shaky ladder hyperplane-defined functions. In: Proceedings of the 2005 genetic and evolutionary computation conference. ACM Press, New York, pp 32–38
38. Jen E (2003) Stable or robust? What's the difference? Complexity 8 3:12–18
39. Bosman PAN (2005) Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Evolutionary algorithms for dynamic optimization problems EvoDOP workshop at the 2005 genetic and evolutionary computation conference, pp 39–47
40. Doyle JC, Wall JE, Stein G (1982) Performance and robustness analysis for structured uncertainty. In: Proceedings of the 21st IEEE conference on decision and control, vol 21, pp 629–636
41. Balas GJ, Doyle JC (1990) Robustness and performance tradeoffs in control design for flexible structures. In: Proceedings of the 29th IEEE conference on decision and control, vol 6, pp 2999–3010
42. Hashtrudi-Zaad K, Salcudean SE (2000) Analysis and evaluation of stability and performance robustness for teleoperation control architectures. In: Proceedings of 2000 IEEE international conference on robotics and automation, vol 4, pp 3107–3113
43. Adivikolanu S, Zafiriou E (2000) Extensions and performance/robustness tradeoffs of the EWMA run-to-run controller by using the internal model control structure. IEEE Trans Electron Packaging Manuf 23(1):56–68
44. Zhou K, Ren Z (2001) A new controller architecture for high performance, robust, and fault-tolerant control. IEEE Trans Autom Control 46(10):1613–1618
45. Grefenstette JJ (1999) Evolvability in dynamic fitness landscapes: a gentic algorithm approach. In: Proceedings of the 1999 IEEE congress on evolutionary computation, vol 3, pp 2031–2038
46. Morrison RW, De Jong KA (1999) A test problem generator for nonstationary environments. In: Proceedings of the 1999 IEEE congress on evolutionary computation, vol 3, pp 2047–2053
47. Yang S, Yao X (2005) Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput 9(11):815–834
48. Mitchell M, Forrest S, Holland JH (1991) The royal road for genetic algorithms: fitness landscapes and GA performance. In: Proceedings of the 1st European conference on artifical life. MIT Press, Cambridge, MA, USA, pp 245–254
49. Whitley LD (1991) Fundamental principles of deception in genetic search. Foundations of Genetic Algorithms 1. Morgan Kaufmann Publishers, pp 221–241
50. Garnier J, Kallel L, Schoenauer M (1999) Rigorous hitting times for binary mutations. Evol Comput 7(2):173–203
51. Jansen T, Wegener I (2001) Evolutionary algorithms—how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Trans Evol Comput 5(6):589–599
52. Ong YS, Lim MH, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybernet Part B 36(1):141–152