Hyper-Selection in Dynamic Environments

Shengxiang Yang and Renato Tinós

***Abstract*— In recent years, several approaches have been developed for genetic algorithms to enhance their performance in dynamic environments. Among these approaches, one kind of methods is to adapt genetic operators in order for genetic algorithms to adapt to a new environment. This paper investigates the effect of the selection pressure on the performance of genetic algorithms in dynamic environments. A hyper-selection scheme is proposed for genetic algorithms, where the selection pressure is temporarily raised whenever the environment changes. The hyper-selection scheme can be combined with other approaches for genetic algorithms in dynamic environments. Experiments are carried out to investigate the effect of different selection pressures on the performance of genetic algorithms in dynamic environments and to investigate the effect of the hyper-selection scheme on the performance of genetic algorithms in combination with several other schemes in dynamic environments. The experimental results indicate that the effect of the hyper-selection scheme depends on the problem under consideration and other schemes combined in genetic algorithms.**

I. Introduction

Dynamic optimization problems (DOPs) are pervasive in the real world since many optimization problems involve dynamic environments. For example, in manufacturing, new jobs may arrive stochastically and machines may break down. The nature of DOPs challenges traditional optimization algorithms because DOPs require them to track the changing environment with time. For DOPs, genetic algorithms (GAs) are a good choice because they are inspired from the principles of biological evolution, which takes place in a dynamic environment in nature. However, when solving DOPs, traditional GAs face a big problem: once converged, GAs are unable to adapt to the new environment when a change occurs. In order to enhance the performance of GAs in dynamic environments, several approaches have been developed in the literature in recent years [2], [7].

Generally speaking, there are four types of approaches for GAs to address DOPs. The first type, called *diversity schemes*, addresses the convergence problem directly by inserting random immigrants [6] or guided immigrants [18], [19] into the population during the run of GAs. The second type of approaches uses *memory*, either implicit [5], [8], [12] or explicit [1], [10], [15], [16], [21], to store and reuse useful information in order to efficiently adapt GAs to returned environments. The third type uses *multi-population* [3], [13] or *speciation* [14] schemes to distribute the search forces into the search space. The fourth type is the *adaptive* scheme that adjusts genetic operators and/or relevant parameters to adapt GAs to the new environment whenever a change occurs, e.g, the hypermutation scheme [4], [11].

Of the four types of approaches devised for GAs for DOPs, the fourth type of adaptive schemes has received relatively less attention so far. This paper investigates the effect of the selection pressure on the performance of GAs in dynamic environments. A *hyper-selection* scheme is proposed for GAs to address DOPs, where the selection pressure is temporarily raised when the environment changes. The hyper-selection scheme can be combined with other schemes in the literature for GAs in dynamic environments. Using the dynamic problem generator proposed in [17], [20], a series of DOPs are constructed as the dynamic test environments and experiments are carried out to investigate the performance of GAs with different selection pressures and the performance of GAs with the hyper-selection scheme in combination with several other schemes for DOPs. Based on the experimental results, the effect of the selection pressure and the hyper-selection scheme on the performance of GAs in dynamic environments is analysed.

The rest of this paper is organized as follows. The next section describes the hyper-selection scheme and several GAs that integrate this scheme and several other schemes, which are studied in this paper. Section III presents the experimental design, including the dynamic test environments, parameter settings, and performance measure. Section IV presents the experimental results and analysis. Finally, Section V concludes this paper with discussions on relevant future work.

II. Description of Algorithms Investigated

A. The Hyper-Selection Scheme

The standard GA maintains and evolves a population of candidate solutions through selection and variation. New populations are generated by first probabilistically selecting relatively fitter individuals from the current population and then performing variation operations, e.g., crossover and mutation, on them to create new offspring. This process repeats until some termination condition is met, e.g., the maximum allowable number of generations t_{max} is reached.

The pseudo-code for the standard GA studied in this paper, which is denoted *SGA*, is shown in Fig. 1, where p_c and p_m are the probability of applying crossover and mutation respectively. In SGA and other GAs studied in this paper, the tournament selection scheme is used for selecting individuals for reproduction. For the tournament selection scheme, when selecting an individual into the mating pool $P'(t)$, we first randomly pick ts individuals from $P(t)$ and then select the best one as the winner to fill $P'(t)$. Here, the parameter ts,

Shengxiang Yang is with the Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, United Kingdom (email: s.yang@mcs.le.ac.uk).

Renato Tinós is with the Department of Physics and Mathematics, FFCLRP, University of São Paulo (USP), 14040-901, Ribeirão Preto, SP, Brazil (email: rtinos@ffclrp.usp.br).

This work was supported by UK EPSRC under Grant No. EP/E060722/1 and Brazil FAPESP under Grant Proc. 04/04289-6.

978-1-4244-1823-7/08/$25.00©2008 IEEE

```
t := 0 and initialize population P(0) randomly
repeat
  evaluate population P(t)
  P'(t) := tournamentSelect(P(t), ts)
  crossover(P'(t), p_c)  // p_c is the crossover prob.
  mutate(P'(t), p_m)     // p_m is the mutation prob.
  replace elites from P(t) into P'(t) randomly
  P(t+1) := P'(t)
  t := t+1
until a termination condition is met    // e.g., t > t_max
```

Fig. 1. Pseudo-code for the standard GA (SGA) with the tournament selection, where ts is the tournament size.

called the *tournament size*, controls the selection pressure. The bigger the value of ts, the higher the selection pressure. In SGA and other GAs studied in this paper, the elitism is applied with size 2. That is, the best two individuals in generation $t-1$ are inherited and replace two random individuals in the current population $P'(t)$ after variation operations. The elites are updated every generation. That is, the best two individuals generated at generation t are taken as the elites at generation t, no matter whether they are better than the elites at generation $t-1$ or not.

Usually, with the running of SGA, individuals in the population will eventually converge to the optimal solution(s) in stationary environments due to the selection pressure. However, in dynamic environments, convergence becomes a big problem for GAs since it deprives the population of genetic diversity and hence make it hard for GAs to adapt to the new environment when a change occurs. To address the convergence problem, several approaches have been developed to maintain the level of population diversity or re-introduce diversity after a change occurs.

However, using only these diversity schemes may not adapt GAs to a new environment to its best. For example, for re-introduced diversity, we may need to apply a high selection pressure to out-stand those really useful new blood and hence adapt GAs more quickly toward a new environment. This thinking naturally leads to the introduction of the *hyper-selection* scheme: whenever an environmental change occurs, the selection pressure is temporarily raised for several generations from the basic pressure.

Obviously, to realize its best advantage, the hyper-selection scheme should be combined with other diversity approaches for GAs to address DOPs. In this paper, the hyper-selection scheme is combined with restart, hypermutation [4], and elitism-based immigrants [19] schemes for GAs, which are described in the following sub-sections respectively.

B. Hyper-Selection with Restart

Restart is a simple and natural way for GAs to address DOPs. For GAs with the restart scheme, whenever the environment changes, the population is re-initialized. The pseudo-code of the GA with restart, denoted *RSGA* in this paper, is shown in Fig. 2. The corresponding GA with restart and hyper-selection schemes, denoted *RHSGA* in this paper, is also shown in Fig. 2. Within RHSGA, whenever the environment changes, the tournament size ts is raised from the basic low value ts^l to a high value ts^u for the following n_{ts} generations.

```
t := 0 and initialize population P(0) randomly
repeat
  evaluate population P(t)
  P'(t) := tournamentSelect(P(t), ts)
  crossover(P'(t), p_c)      // p_c is the crossover prob.
  mutate(P'(t), p_m)         // p_m is the mutation prob.
  replace elites from P(t) into P'(t) randomly

  if the environment changes then
    re-initialize P'(t)
    if hyper-selection used then    // for RHSGA
      raise ts from ts^l to ts^u for n_hs generations

  P(t+1) := P'(t)
  t := t+1
until a termination condition is met    // e.g., t > t_max
```

Fig. 2. Pseudo-code for the GA with re-start (RSGA) and the GA with restart and hyper-selection (RHSGA).

```
t := 0 and initialize population P(0) randomly
repeat
  evaluate population P(t)
  P'(t) := tournamentSelect(P(t), ts)
  crossover(P'(t), p_c)      // p_c is the crossover prob.
  mutate(P'(t), p_m)         // p_m is the mutation prob.
  replace elites from P(t) into P'(t) randomly

  if the environment changes then
    raise p_m from p_m^l to p_m^u for n_hm generations
    if hyper-selection used then    // for HMSGA
      raise ts from ts^l to ts^u for n_hs generations

  P(t+1) := P'(t)
  t := t+1
until a termination condition is met    // e.g., t > t_max
```

Fig. 3. Pseudo-code for the GA with hypermutation (HMGA) and the GA with hypermutation and hyper-selection (HMSGA).

C. Hyper-Selection with Hypermutation

Hypermutation is another scheme to re-introduce the population diversity for GAs to address DOPs. The pseudo-code of the GA with hypermutation, denoted *HMGA*, is shown in Fig. 3. Within HMGA, whenever the environment changes, the mutation probability p_m is raised from the basic low value p_m^l to a high value p_m^u for the following n_{hm} generations. The corresponding GA with hypermutation and hyper-selection schemes, denoted *HMSGA* in this paper, is also shown in Fig. 3.

$t := 0$ and initialize population $P(0)$ randomly
repeat
  evaluate population $P(t)$
  // perform elitism-based immigration
  denote the elite in $P(t)$ by $E(t)$
  generate $r_{ei} \times n$ immigrants by mutating $E(t)$
  evaluate these elitism-based immigrants
  replace worst individuals in $P'(t)$ by these immigrants

  $P'(t) :=$ tournamentSelect$(P(t), ts)$
  crossover$(P'(t), p_c)$ *// p_c is the crossover prob.*
  mutate$(P'(t), p_m)$ *// p_m is the mutation prob.*

  if the environment changes **then** *// for EIHSGA*
    raise ts from ts^l to ts^u for n_{hs} generations

  $P(t+1) := P'(t)$
  $t := t + 1$
until a termination condition is met *// e.g., $t > t_{max}$*

Fig. 4. Pseudo-code for the GA with the elitism-based immigrants scheme (EIGA) and the GA with the elitism-based immigrants and hyper-selection schemes (EIHSGA).

D. Hyper-Selection with Elitism-based Immigrants

Another approach that addresses the convergence problem of GAs for DOPs is to use immigrants to maintain the diversity level of the population. For example, Grefenstette proposed the random immigrants [6], which works by replacing random individuals into the population. This benefits the performance of GAs in dynamic environments, especially when a change occurs.

Recently, a guided immigrants scheme, called *elitism-based immigrants*, has been proposed for GAs to address DOPs in [19]. The elitism-based immigrants scheme combines the idea of elitism with random immigrants scheme. The pseudo-code for the GA with the elitism-based immigrants scheme, denoted *EIGA* in this paper, is shown in Fig. 4. Within EIGA, for each generation t, the elite $E(t)$ from the previous generation is used as the base to create immigrants. From $E(t)$, a set of $r_{ei} \times n$ individuals are iteratively generated by mutating $E(t)$ bitwise with a probability p_m^i, where n is the population size and r_{ei} is the ratio of the number of elitism-based immigrants to the population size. The generated individuals then replace the worst individuals in the current population.

The elitism-based immigrants scheme has been experimentally validated to be efficient to improve the performance of GAs in dynamic environments [19]. In this paper, in order to investigate the effect of the hyper-selection scheme on the performance of GAs with the elitism-based immigrants scheme, we integrate the hyper-selection scheme into EIGA. The pseudo-code for the GA with hyper-selection and elitism-based immigrants schemes, denoted *EIHSGA* in this paper, is also shown in Fig. 4.

III. Experimental Design

A. Dynamic Test Environments

The DOP generator proposed in [17], [20] is used to construct dynamic test environments for this study. This generator can construct DOPs from any binary-encoded stationary function $f(\vec{x})$ as follows. Suppose the environment changes every τ generations. For each environment k, an XORing mask $\vec{M}(k)$ is incrementally generated as follows:

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

where "$\oplus$" is a bitwise exclusive-or (XOR) operator and $\vec{T}(k)$ is an intermediate binary template for environment k. $\vec{T}(k)$ is created with $\rho \times l$ ($\rho \in (0.0, 1.0]$) random loci set to 1 while the remaining loci set to 0. For the first initial environment $k = 1$, $\vec{M}(1)$ is set to a zero vector.

An individual at generation t is evaluated as follows:

$$f(\vec{x}, t) = f(\vec{x} \oplus \vec{M}(k)), \tag{2}$$

where $k = \lceil t/\tau \rceil$ is the environmental index at time t. With this XOR DOP generator, τ and ρ control the speed and severity of environmental changes respectively. Smaller τ means faster changes while bigger ρ means severer changes.

In this paper, three 100-bit binary functions are selected as the base stationary functions to construct dynamic test environments. The first one is the well-known *OneMax* function that aims to maximize the number of ones in a binary string. The second one is a variant of Forrest and Mitchell's *Royal Road* function [9], which consists of 25 contiguous 4-bit building blocks (BBs). Each BB of *Royal Road* contributes 4 to the total fitness if all bits inside the BB have the allele of one; otherwise, it contributes 0. The third problem is a 100-item 0-1 knapsack problem with the weight and profit of each item randomly created in the range of $[1, 30]$ and the capacity of the knapsack set to be half of the total weight of all items. The fitness of a feasible solution is the sum of the profits of the selected items. If a solution overfills the knapsack, its fitness is set to the difference between the total weight of all items and the weight of selected items, multiplied by a small factor 10^{-5} in order to make it in-competitive with those solutions that do not overfill the knapsack.

Dynamic environments are constructed from each of the three base functions using the aforementioned XOR DOP generator. For each dynamic environment, the landscape is periodically changed every τ generations during the run of a GA. In order to compare the performance of GAs in different dynamic environments, the speed of change parameter τ is set to 20, 50, and 100 respectively. The severity of change parameter ρ is set to 0.1, 0.2, 0.5, 0.9, 1.0, and random values in the range of $[0.0, 1.0]$ (i.e., $\rho = rand(0.0, 1.0)$ for each environmental change) respectively.

B. Parameter Settings and Performance Measure

Two sets of experiments were carried out in this paper on the above constructed dynamic test environments. The first set of experiments investigates the effect of the selection

pressure on the performance of SGA for DOPs. The second set investigates the effect of the hyper-selection scheme on the performance of several GAs with certain enhancements from the literature, as described in Section II.

For all GAs, some common parameters are set as follows: generational, 2-point crossover with $p_c = 0.6$, bit flip mutation, tournament selection, and elitism of size 2. For the first set of experiments, the parameters for SGA are set as follows: the population size $n = 120$, mutation probability $p_m = 0.01$, and tournament size ts is set to 2, 6, and 10 respectively (and the SGA is denoted as *ts-SGA* accordingly). For the second set of experiments, the parameters are set as follows. The tournament size for the tournament selection is fixed to $ts = 2$ for RSGA, HMGA, and EIGA and is set to the base value $ts = ts^l = 2$ for normal generations or the hyper value $ts = ts^u = 10$ for the interim generations when the hyper-selection scheme is triggered for RHSGA, HMSGA, and EIHSGA. The population size is set to $n = 120$ for RSGA, RHSGA, HMGA, and HMSGA, and is set to $n = 100$ for EIGA and EIHSGA. The immigrants ratio for EIGA and EIHSGA is set to $r_{ei} = 0.2$. Hence, each GA has 120 evaluations per generation. For EIGA and EIHSGA, $p_m^i = 0.01$. For HMGA and HMSGA, $p_m^l = 0.01$ and $p_m^u = 0.3$. Whenever the environment changes, the hyper-mutation and/or hyper-selection schemes for GAs are triggered for 5 generations, i.e., $n_{hs} = 5$ and $n_{hm} = 5$.

For each experiment of a GA on a DOP, 30 independent runs were executed with the same set of random seeds. For each run, 50 environmental changes were allowed. For each run the best-of-generation fitness was recorded every generation. The overall performance of an algorithm on a DOP is defined as:

$$\overline{F}_{BOG} = \frac{1}{G}\sum_{i=1}^{G}(\frac{1}{30}\sum_{j=1}^{30}F_{BOG_{ij}}), \tag{3}$$

where $G = 50*\tau$ is the total number of generations for a run and $F_{BOG_{ij}}$ is the best-of-generation fitness of generation i of run j. The off-line performance $\overline{F}_{BOG}$ is the best-of-generation fitness averaged over 30 runs and then averaged over the data gathering period.

IV. Experimental Results and Analysis

A. Experimental Results on Selection Pressure

The experimental results of the first set of experiments are plotted in Fig. 5. The corresponding statistical results of comparing GAs by one-tailed t-test with 58 degrees of freedom at a 0.05 level of significance are given in Table I. The t-test result with respect to $Alg.\ 1 - Alg.\ 2$ is shown as "+", "−", "$s+$", or "$s-$" when $Alg.\ 1$ is better than, worse than, significantly better than, or significantly worse than $Alg.\ 2$ respectively. From Fig. 5 and Table I, the following two results can be observed.

First, it can be seen that the selection pressure does have a significant effect on the performance of SGA on most dynamic test problems. This result can be clearly seen from Table I, where most t-test results are shown as either "$s+$" or "$s-$".

Second, the exact effect of increasing the selection pressure on the performance of SGA depends on the base function used for DOPs. The effect is quite different across dynamic OneMax, Royal Road and Knapsack problems. For dynamic OneMax problems, it seems that a higher selection pressure degrades the performance of SGA except for severely changing environments with $\rho = 0.9$ or 1.0. For dynamic Royal Road problems, it seems that a higher selection pressure is always beneficial for the performance of SGA, see the t-test results regarding 6-SGA − 2-SGA and 10-SGA − 6-SGA. On the dynamic Knapsack problems, when the selection pressure is increased from $ts = 2$ to $ts = 6$, the performance of SGA improves on many DOPs. But, when the selection pressure is further increased from $ts = 6$ to $ts = 10$, the performance of SGA is degraded on most DOPs.

B. Experimental Results on Hyper-Selection

The experimental results of the second set of experiments regarding the effect of the hyper-selection scheme are plotted in Fig. 6. The corresponding statistical results of comparing GAs by one-tailed t-test with 58 degrees of freedom at a 0.05 level of significance are given in Table II. In order to better understand the performance of GAs, the dynamic behaviour of GAs with respect to the best-of-generation fitness against generations on DOPs with $\tau = 50$ and $\rho = 0.1$ and $\rho = 0.9$ is plotted in Fig. 7. In Fig. 7, the first 10 environmental changes, i.e., 500 generations, are shown and the data were averaged over 30 runs. From Figs. 6 and 7 and Table II, several results can be observed.

First, regarding the restart scheme, it can be seen that the performance of RSGA and RHSGA is not sensitive to the value of ρ. For example, on the dynamic Royal Road functions with $\tau = 50$ and $\rho = 0.1$, 0.2, 0.5, 0.9, 1.0, and $rand$, the performance of RSGA is $\overline{F}_{BOG}(\text{RSGA}) = 40.8$, 40.9, 40.9, 40.9, 41.0, and 40.9 respectively. This result is easy to understand since each time the environment changes, RSGA and RHSGA are in fact put in the same starting point for the same base function given that the population is re-initialized. This result can be further seen from the dynamic behaviour of RSGA and RHSGA in Fig. 7, where their behaviour for each environment is almost the same.

It can also be seen that RHSGA significantly outperforms RSGA on all DOPs, see the t-test results regarding RHSGA − RSGA in Table II. For example, on the dynamic Royal Road functions with $\tau = 50$ and $\rho = 0.1$, 0.2, 0.5, 0.9, 1.0, and $rand$, the performance of RHSGA is $\overline{F}_{BOG}(\text{RHSGA}) = 43.8$, 43.7, 43.7, 43.7, 43.8, and 43.7 respectively. This result shows the benefit of the hyper-selection scheme: temporarily raising the selection pressure always helps improve the performance of SGA with restart in dynamic environments.

Second, regarding the hypermutation scheme, it can be seen that the performance of HMGA and HMSGA is now sensitive to the value of ρ. Their performance drops when the value of ρ increases from 0.1 to 0.2 to 0.5. When the value

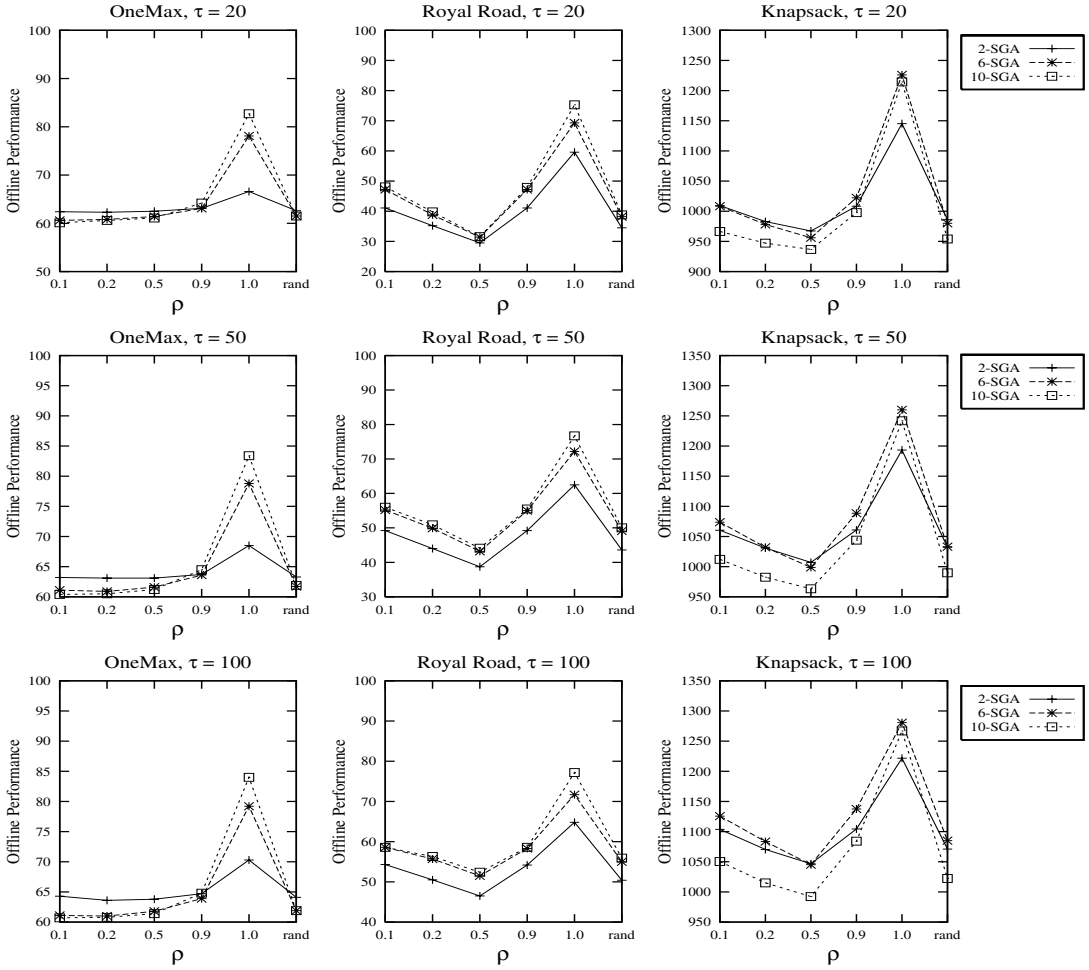

Fig. 5. Experimental results of comparing SGAs with different tournament size ts on dynamic test problems.

TABLE I

THE t-TEST RESULTS OF COMPARING SGAS WITH DIFFERENT TOURNAMENT SIZE ts ON DOPS.

t-test Result	*OneMax*						*Royal Road*						*Knapsack*					
$\tau = 20$, $\rho \Rightarrow$	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand
6-SGA − 2-SGA	$s-$	$s-$	$s-$	$+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$-$	$s-$	$s-$	$s+$	$s+$	$s-$
10-SGA − 2-SGA	$s-$	$s-$	$s-$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s+$	$s-$
10-SGA − 6-SGA	$s-$	$s-$	$s-$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
$\tau = 50$, $\rho \Rightarrow$	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand
6-SGA − 2-SGA	$s-$	$s-$	$s-$	$-$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$+$	$s-$	$s+$	$s+$	$+$
10-SGA − 2-SGA	$s-$	$s-$	$s-$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s+$	$s-$
10-SGA − 6-SGA	$s-$	$s-$	$s-$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
$\tau = 100$, $\rho \Rightarrow$	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand
6-SGA − 2-SGA	$s-$	$s-$	$s-$	$s-$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$-$	$s+$	$s+$	$s+$
10-SGA − 2-SGA	$s-$	$s-$	$s-$	$+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s+$	$s-$
10-SGA − 6-SGA	$s-$	$s-$	$s-$	$s+$	$s+$	$-$	$+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$

of ρ is further increased to 0.9 and 1.0, their performance rises instead of drops.

It can also be seen that the effect of adding the hyper-selection scheme to HMGA is positive on dynamic Royal Road functions but negative on dynamic OneMax and Knapsack problems, see the t-test results regarding HMSGA − HMGA in Table II.

Third, when considering the elitism-based immigrants scheme, it can be seen that both EIGA and EIHSGA clearly outperforms GAs with restart and hypermutation schemes,

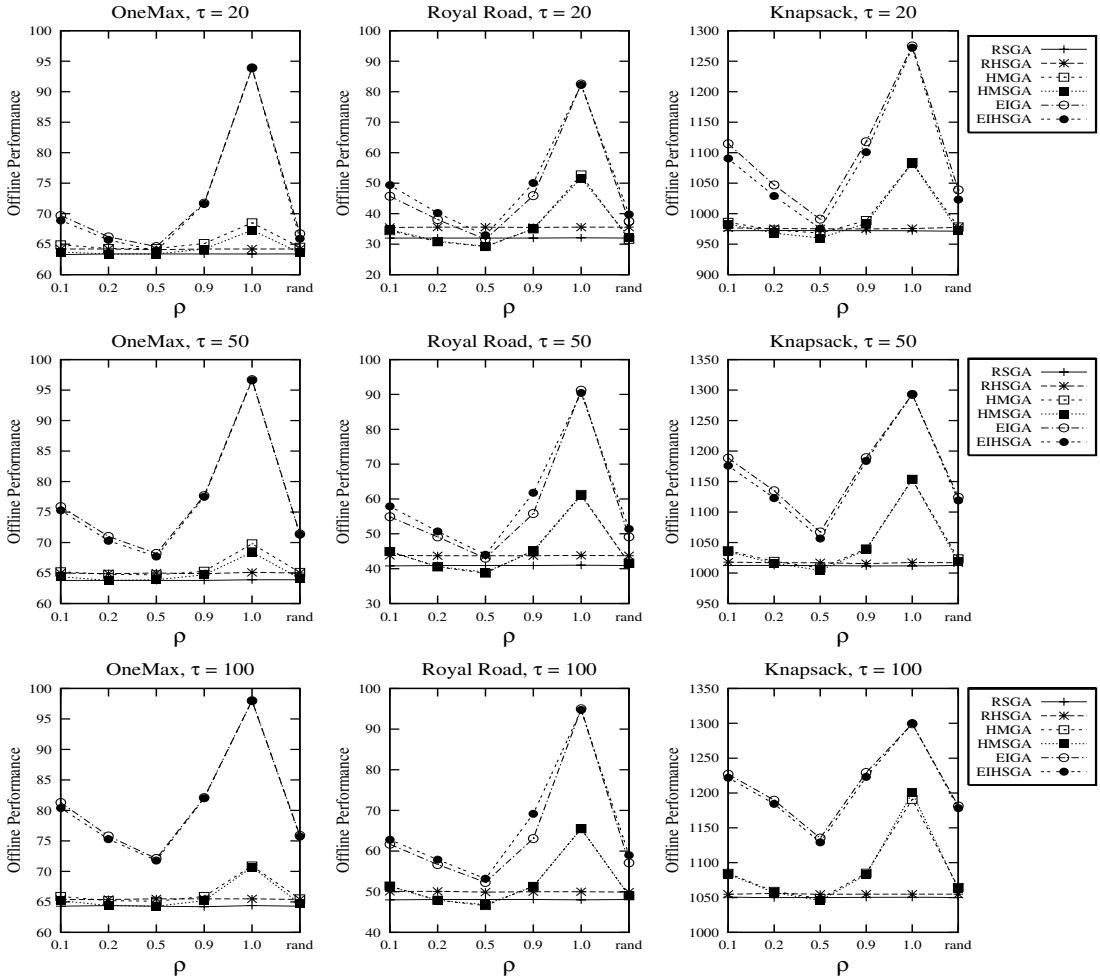

Fig. 6. Experimental results of comparing GAs with and without hyper-selection on dynamic test problems.

TABLE II

THE t-TEST RESULTS OF COMPARING GAS WITH AND WITHOUT HYPER-SELECTION ON DOPS.

t-test Result	*OneMax*						*Royal Road*						*Knapsack*					
$\tau = 20$, $\rho \Rightarrow$	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand
RHSGA − RSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
HMSGA − HMGA	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s+$	$s+$	$+$	$+$	$-$	$s+$	$s-$	$s-$	$s-$	$s-$	$-$	$s-$
EIHSGA − EIGA	$s-$	$s-$	$s-$	$-$	$-$	$s-$	$s+$	$s+$	$s+$	$s+$	$-$	$s+$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
EIHSGA − RHSGA	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$
EIHSGA − HMSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$\tau = 50$, $\rho \Rightarrow$	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand
RHSGA − RSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
HMSGA − HMGA	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$+$	$s+$	$s+$	$s-$	$-$	$+$	$-$	$s-$	$s-$	$-$	$-$	$s-$
EIHSGA − EIGA	$s-$	$s-$	$s-$	$-$	$-$	$-$	$s+$	$s+$	$s+$	$s+$	$-$	$s+$	$s-$	$s-$	$s-$	$s-$	$+$	$s-$
EIHSGA − RHSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIHSGA − HMSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$\tau = 100$, $\rho \Rightarrow$	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand	0.1	0.2	0.5	0.9	1.0	rand
RHSGA − RSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
HMSGA − HMGA	$s-$	$s-$	$s-$	$s-$	$-$	$s-$	$-$	$-$	$s+$	$-$	$+$	$+$	$-$	$-$	$s-$	$-$	$s+$	$-$
EIHSGA − EIGA	$s-$	$s-$	$s-$	$-$	$-$	$-$	$s+$	$s+$	$s+$	$s+$	$-$	$s+$	$s-$	$s-$	$s-$	$s-$	$+$	$s-$
EIHSGA − RHSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIHSGA − HMSGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$

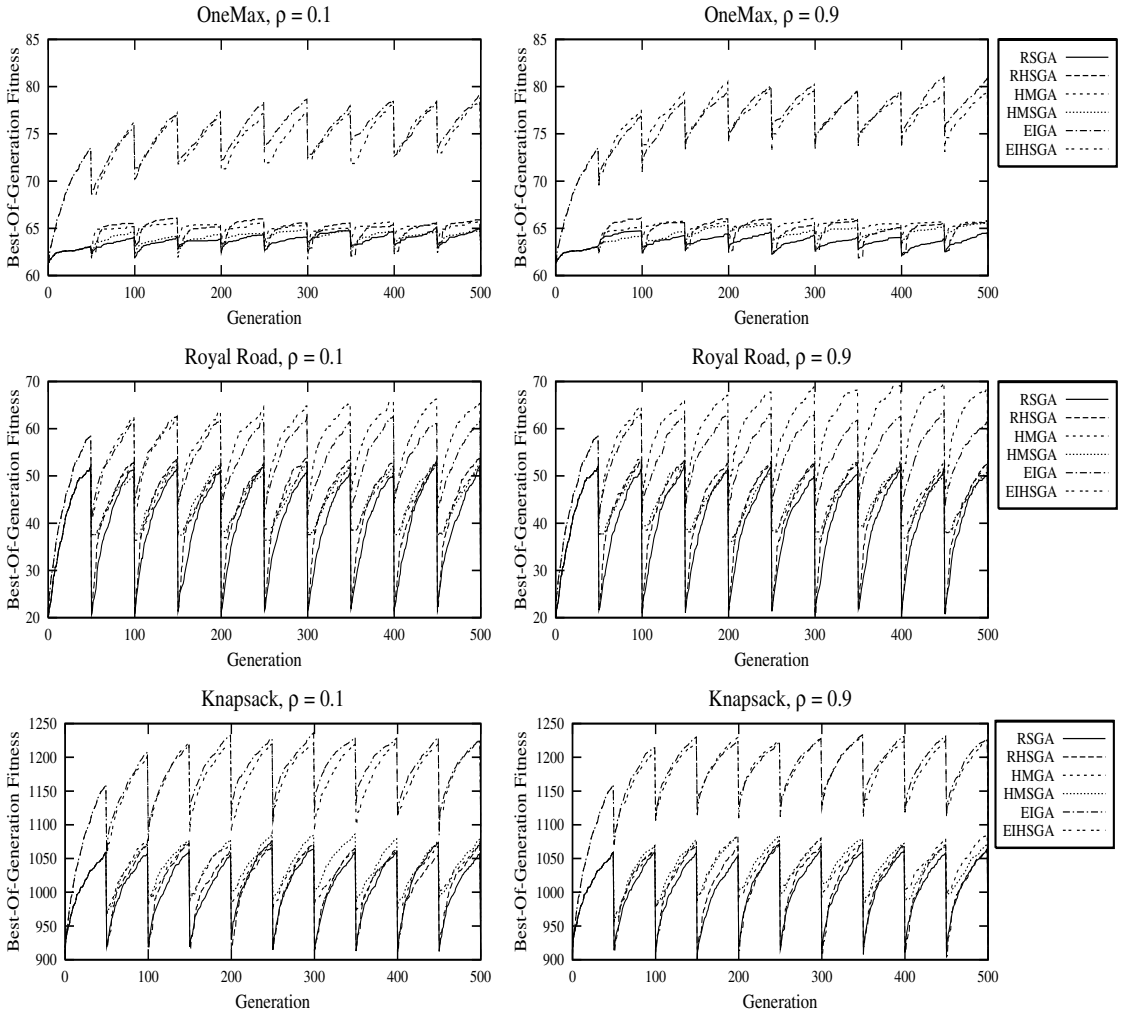

Fig. 7. Dynamic behaviour of GAs on DOPs with $\tau = 50$ and $\rho = 0.1$ (Left) and $\rho = 0.9$ (Right).

see the t-test results regarding EIHSGA $-$ RHSGA and EIHSGA $-$ HMSGA in Table II. This result can also be observed from the dynamic behaviour of GAs in Fig. 7. Both EIGA and EIHSGA maintain a much higher fitness level than other GAs do. The reason to this result lies in that the elitism-based immigrants scheme is more efficient for GAs than the restart and hypermutation schemes.

Another observation is that there is no clear winner between EIGA and EIHSGA on the test DOPs. The hyper-selection scheme is beneficial for the performance of EIHSGA on dynamic Royal Road problems but degrades the performance of EIHSGA on dynamic OneMax and Knapsack problems.

Finally, comparing the performance of GAs in Fig. 6 with the performance of 2-SGA in Fig. 6, it can be seen that the restart, hypermutation, and elitism-based immigrants schemes do improve the performance of SGAs on many DOPs. But, there are also quite some cases where 2-SGA outperforms GAs with restart or hypermutation, especially when the value of ρ is small. For example, on the dynamic Royal Road function with $\tau = 50$ and $\rho = 0.1$, the performance of 2-SGA, RSGA, and HMGA is $\overline{F}_{BOG}(\text{2-SGA}) = 49.2$, $\overline{F}_{BOG}(\text{RSGA}) = 40.8$, and $\overline{F}_{BOG}(\text{HMGA}) = 44.9$ respectively. This result warns us that when the environment changes slightly, a high diversity introduced may divert the searching force too much and hence degrades the performance of GAs.

V. Conclusions

Adapting genetic operators is one type of approaches for GAs to address dynamic environments. This paper investigates the effect of the selection pressure on the performance of GAs in dynamic environments and proposes a hyper-selection scheme for GAs to solve DOPs. When

an environmental change occurs, the selection pressure can be temporarily raised. The hyper-selection scheme can be combined with other schemes in the literature for GAs in dynamic environments.

The effect of selection pressure and the hyper-selection scheme for GAs in dynamic environments were experimentally studied based on a series of constructed dynamic test problems. From the experimental results and relevant analysis, three major conclusions can be drawn on the dynamic test environments. First, the selection pressure does have an important effect on the performance of GAs in dynamic environments. Second, the effect of increasing the selection pressure on the performance of SGAs in dynamic environments is problem dependent. Third, the effect of the hyper-selection scheme on the performance of GAs in dynamic environments depends on the problem being solved and other approaches used in GAs.

Generally speaking, this paper for the first time investigates the effect of selection pressure for GAs in dynamic environments with some preliminary experiments. The results observed can be used to guide the design of new GAs for DOPs. For example, developing more efficient selection schemes that can adjust the selection pressure adaptively during the running of GAs may be an interesting future work. Combining the hyper-selection scheme with other mechanisms for GAs in dynamic environments is another interesting future work.

REFERENCES

[1] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. *Proc. of the 1999 IEEE Congress on Evol. Comput.*, vol. 3, pp. 1875–1882, 1999.

[2] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

[3] J. Branke, T. Kaußler, C. Schmidth, and H. Schmeck. A multi-population approach to dynamic optimization problems. *Proc. of the 4th Int. Conf. on Adaptive Computing in Design and Manufacturing*, pp. 299–308, 2000.

[4] H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. *Proc. of the 5th Int. Conf. on Genetic Algorithms*, pp. 523–530, 1993.

[5] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, pp. 59–68, 1987.

[6] J. J. Grefenstette. Genetic algorithms for changing environments. *Parallel Problem Solving from Nature II*, pp. 137–144, 1992.

[7] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments: a survey. *IEEE Trans. on Evol. Comput.*, vol. 9, no. 3, pp. 303–317, June 2005.

[8] J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. *Proc. of the 4th Int. Conf. on Parallel Problem Solving from Nature*, pp. 139–148, 1998.

[9] M. Mitchell, S. Forrest and J. H. Holland. The royal road for genetic algorithms: fitness landscapes and GA performance. *Proc. of the 1st European Conf. on Artificial Life*, pp. 245–254, 1992.

[10] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. *Proc. of the 7th Int. Conf. on Genetic Algorithms*, pp. 299-306, 1997.

[11] R. W. Morrison and K. A. De Jong. Triggered hypermutation revisited. *Proc. of the 2000 IEEE Congress on Evolutionary Computation*, pp. 1025-1032, 2000.

[12] K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimisation. *Proc. of the 6th Int. Conf. on Genetic Algorithms*, 1995.

[13] F. Oppacher and M. Wineberg. The shifting balance genetic algorithm: Improving the GA in a dynamic environment. *Proc. of the 1999 Genetic and Evolutionary Computation Conference*, vol. 1, pp. 504-510, 1999.

[14] D. Parrott and X. Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Trans. on Evol. Comput.*, vol. 10, no. 4, pp. 444-458, 2006.

[15] A. Simões and E. Costa. An immune system-based genetic algorithm to deal with dynamic environments: diversity and memory. *Proc. of the 6th Int. Conf. on Neural Networks and Genetic Algorithms*, pp. 168-174, 2003.

[16] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. *Proc. of the 1999 Congress on Evol. Comput.*, pp. 1843–1850, 1999.

[17] S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithm. *Proc. of the 2003 Congress on Evol. Comput.*, vol. 3, pp. 2246-2253, 2003.

[18] S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. *Proc. of the 2005 Genetic and Evol. Comput. Conference*, vol. 2, pp. 1115-1122, 2005.

[19] S. Yang. Genetic algorithms with elitism-based immigrants for changing optimization problems. *Applications of Evolutionary Computing*, LNCS 4448, pp. 627-636, 2007.

[20] S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, vol. 9, no. 11, pp. 815-834, 2005.

[21] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans. on Evol. Comput.*, to appear, 2008.