Chapter 6
Memetic Algorithms for Dynamic Optimization Problems

Hongfeng Wang and Shengxiang Yang

Abstract. Dynamic optimization problems challenge traditional evolutionary algorithms seriously since they, once converged, cannot adapt quickly to environmental changes. This chapter investigates the application of memetic algorithms, a class of hybrid evolutionary algorithms, for dynamic optimization problems. An adaptive hill climbing method is proposed as the local search technique in the framework of memetic algorithms, which combines the features of greedy crossover-based hill climbing and steepest mutation-based hill climbing. In order to address the convergence problem, a new immigrants scheme, where the immigrant individuals can be generated from mutating an elite individual adaptively, is also introduced into the proposed memetic algorithm for dynamic optimization problems. Based on a series of dynamic problems generated from several stationary benchmark problems, experiments are carried out to investigate the performance of the proposed memetic algorithm in comparison with some peer algorithms. The experimental results show the efficiency of the proposed memetic algorithm in dynamic environments.

6.1 Introduction

Many real-world optimization problems are dynamic optimization problems (DOPs), where the function landscapes may change over time and, thus, the optimum of these problems may also change over time. DOPs require powerful heuristics that account for the uncertainty present in the real world. Since evolutionary algorithms (EAs)

Hongfeng Wang
College of Information Science and Engineering, Northeastern University,
Shenyang 110004, China
e-mail: hfwang@mail.neu.edu.cn

Shengxiang Yang
Centre for Computational Intelligence (CCI), School of Computer Science and Informatics,
De Montfort University, The Gateway, Leicester LE1 9BH, U.K.
e-mail: syang@dmu.ac.uk

draw their inspiration from the principles of natural evolution, which is a stochastic and dynamic process, they also seem to be suitable for DOPs. However, traditional EAs face a serious challenge for DOPs because they cannot adapt well to the changing environment once converged.

In order to address DOPs, many approaches have been developed [46] and can be grouped into five categories: 1) increasing population diversity after a change is detected, such as the adaptive mutation methods [5, 38]; 2) maintaining population diversity throughout the run, such as the immigrants approaches [12, 44]; 3) memory approaches, including implicit [11, 37] and explicit memory [3, 40, 43, 48] methods; 4) multi-population [4, 27] and speciation approaches [30]; 5) prediction methods [2, 24, 31, 32]. A comprehensive survey on EAs applied to dynamic environments can be found in [15, 25, 45]

In recent years, there has been an increasing concern from the evolution computation community on a class of hybrid EAs, called memetic algorithms (MAs), which hybridize local search (LS) methods with EAs to refine the solution quality. So far, MAs have been widely used for solving many optimization problems, such as scheduling problems [14, 20, 21], combinatorial optimization problems [9, 35, 36], multi-objective problems [10, 13, 19] and other applications [49, 50]. However, these problems for which MAs have been applied are mainly stationary problems. MAs have rarely been applied for DOPs [7, 8, 41]. During the running course of general MAs, they may always exhibit very strong exploitation capacity due to executing efficient local refinement on individuals, but they may lose the exploration capacity as a result of the population converging to one optimum, which needs to be avoided in dynamic environments. Therefore, it becomes an interesting research issue to examine the performance of MAs, which are enhanced by suitable diversity methods, for DOPs.

In this chapter, we investigate the application of an MA with an adaptive hill climbing strategy, which combines the features of crossover-based hill climbing and mutation-based hill climbing in both cooperative and competitive fashions, to address DOPs. In order to address the convergence problem, a new immigrants scheme, where three different immigrants schemes are generalized within a uniform framework, is introduced into the proposed MA to improve its performance in dynamic environments. In addition, two different dominated schemes, which are used to keep the balance of extra computation costs between LS and diversity maintaining, are experimentally investigated in the proposed MAs for DOPs.

The rest of this paper is organized as follows. Section 6.2 describes the proposed MA in detail, including the framework of general genetic algorithm (GA)-based MA, the proposed LS operators and immigrants scheme, and discussion on how to determining the computation costs of LS and diversity maintaining. Section 6.3 introduces a series of DOPs generated by a dynamic problem generator from the stationary test suite. Section 6.4 reports the experimental results and relevant analysis. Finally, Section 6.5 concludes this paper with some discussions on relevant future work.

```
Procedure General GA-based MA:
begin
  parameterize();
  $t := 0$;
  initializePopulation($P(0)$);
  evaluatePopulation($P(0)$);
  $E(0)$ := selectForLocalSearch($P(0)$);
  localSearch($E(0)$);
  repeat
    $P'(t)$ := selectForReproduction($P(t)$);
    $P''(t)$ := crossover($P'(t)$);
    mutate($P''(t)$);
    evaluatePopulation($P''(t)$);
    $P(t+1)$ := selectForSurvival($P(t)$, $P''(t)$);
    $E(t)$ := selectForLocalSearch($P(t)$);
    localSearch($E(t)$);
    $t := t+1$;
  until a stop condition is met
end
```

Fig. 6.1 Pseudo-code for a general GA-based MA

6.2 Investigated Algorithms

6.2.1 Framework of GA-Based Memetic Algorithms

The MAs investigated in this paper are a class of GA-based MAs, which can be expressed by the pseudo-code shown in Fig. 6.1. Within these MAs, a population $P(0)$ of *pop_size* individuals are generated randomly and then evaluated at the initialization step. Then, a set $E(0)$ of individuals are selected from $P(0)$ to be improved by LS. At each subsequent generation, individuals are selected randomly or proportionally from the current population and undergo the uniform crossover operation with a crossover probability. Uniform crossover is the generalization of n-point crossover which creates offspring by deciding, for each bit of one parent, whether to swap the allele of that bit with the corresponding allele of the other parent. After crossover is executed, the bit-wise mutation operator is performed for each newly generated offspring individual, which may change the allele in each locus of an offspring bitwise (0 to 1 and vice versa) with a mutation probability. Then, the *pop_size* best individuals among all parents and offspring are selected to proceed into the next generation and a set $E(t)$ of individuals, which are selected from the newly generated population, are improved by the LS strategy.

Obviously, the LS procedure in an MA which is illustrated in the above pseudo-code would include two steps: first, to select individuals from the population P to construct the set E, and then to apply LS operation to each selected individual in E

to refine it. In the following section, we will give the relevant schemes on the LS operator used in this chapter, where the empirical experience and theoretical reasoning are both used in the design. Here, the aim that the above-mentioned pseudo-code is introduced is only to provide a sound basis for understanding the general framework of used MAs in this chapter.

It is more noticeable that some diversity maintaining approaches have to be used in order to address the convergence problem when an MA is applied for DOPs, which always consume extra computational cost since LS operators involve fitness evaluations. We will also discuss the relevant solutions to the problems when diversity schemes are applied and how to keep the balance of the extra computational cost between the diversity schemes and the LS operators, in the later sections.

6.2.2 Local Search

In MAs, EA operations are used for rough global exploration and LS operations are used for directive local refinements to ensure sufficient exploitation during the course of evolving the population. In many relevant researches, LS is applied to each newly generated individual, which would consume a huge number of extra fitness evaluations in the search of higher-quality solution. This traditional scheme seems to be too costly and infeasible for MAs in dynamic environments where the environmental changes can occur with the increment of fitness evaluations. Here, only the best fitness individual $elite$ in the population, which means that the set E only comprises one member, should be executed the local refinement considering that $elite$ would lead the running course of algorithm with a greater degree.

Another primary issue that affects the behavior of LS is which LS operator should be used to improve the quality of an individual. Among many LS methods available in the literature, hill climbing (HC) is a common strategy. The basic idea is to use stochastic iterative HC as the move acceptance criterion of the search (i.e., move the search from the current individual to a candidate individual if the candidate has a better fitness). In the context of GAs, HC methods may be divided into two ways: crossover-based hill climbing and mutation-based hill climbing, depending on whether crossover or mutation is used as the move operator in a local area. Here, we propose two HC methods, a greedy crossover-based HC (GCHC) and a steepest mutation-based HC (SMHC), in this section. They are specially designed for MAs with binary encoding scheme, which are our concern in this chapter. The two HC methods are described as follows.

1) GCHC: In this strategy, the individual (*chr*) selected for local improvement is taken as one parent and another parent is selected from the current population using the roulette wheel selection scheme. Then, a special uniform crossover is executed between these two parent individuals to generate an offspring. The offspring will replace the individual *chr* if it has a better fitness than the latter. This procedure is outlined in Fig. 6.2, where a maximization optimization problem is assumed.

2) SMHC: The steepest mutation means that the chromosome only changes several bits randomly when executing one mutation operation on it. In SMHC, the

```
Procedure GCHC(chr):
begin
  calculate($\xi, pc_{ls}$);
  for $i := 1$ to $num_{ls}$ do
    $par\_chr :=$ selectParentForCrossover($P$);
    for $j := 1$ to $n$ do
      if $random() < pc_{ls}$ then
        $chi\_chr[j] := par\_chr[j]$;
      else
        $chi\_chr[j] := chr[j]$;
    endfor
    evaluate($chi\_chr$);
    if $f(chi\_chr) > f(chr)$ then $chr := chi\_chr$;
  endfor
end

GCHC's denotations:
  $\xi$: a population index used to renew the value of $pc_{ls}$
  $pc_{ls}$: the crossover probability in GCHC
  $par\_chr$: the proportionally selected parent individual
  $n$: individual length (problem dependent)
  $random()$: a random number between 0 and 1
  $chi\_chr$: the new individual generated by performing a uniform crossover operation
    between $par\_chr$ and $chr$
  $chr$: the individual selected for LS
```

Fig. 6.2 Pseudo-code for the GCHC operator

individual (*chr*) being improved by LS is picked out and several random bits are changed. If the newly mutated individual has a better fitness, it will replace the individual *chr*. The SMHC strategy is outlined in Fig. 6.3.

From Fig. 6.2 and Fig. 6.3, it can be seen that two important parameters, pc_{ls} in GCHC and nm_{ls} in SMHC, respectively, may affect their performance. In GCHC the smaller the value of pc_{ls}, the more the offspring inherits from the selected individual *chr*. This means executing one step LS operation in a smaller area around *chr*. Similar results can be obtained for nm_{ls} in SMHC. When the value of nm_{ls} is larger, SMHC will perform the LS operation within a wider range around *chr*.

Therefore, the question that remains to be answered here is how to set the two parameters. Generally speaking, the methods of setting strategy parameters in GAs can be classified into three categories [6]: *deterministic mechanism* where the value of the strategy parameter is controlled by some deterministic rules without any feedback from the search, *adaptive mechanism* where there is some form of feedback information from the search process that is used to direct the setting of a strategy

```
Procedure SMHC($chr$):
begin
   calculate($\xi, nm_{ls}$);
   for $i := 1$ to $num_{ls}$ do
      for $j := 1$ to $n$ do
         $chi\_chr[j] := chr[j]$;
      endfor
      for $k := 1$ to $nm_{ls}$ do
         $loc := random(1,n)$;
         $chi\_chr[loc] := 1 - chi\_chr[loc]$;
      endfor
      evaluate($chi\_chr$);
      if $f(chi\_chr) > f(chr)$ then $chr := chi\_chr$;
   endfor
end;

SMHC's denotations:
   $nm_{ls}$: the number of bits mutated in SMHC
   $chi\_chr$: the new individual generated by performing a steepest mutation operation
      upon $chr$
   $loc$: a random selected location for flipping
   $random(1,n)$: a random integer between 1 and $n$
   Other settings are the same as those for GCHC
```

Fig. 6.3 Pseudo-code for the SMHC operator

parameter, and *self-adaptive mechanism* where the parameter to be adapted is encoded into the chromosomes and undergoes genetic operators.

Two different parameter-setting methods will be discussed for pc_{ls} and nm_{ls} in the later experiments. In the *deterministic* method, both pc_{ls} and nm_{ls} are set to constant values, which means that the LS operation will always be executed in a local area of a certain fixed range. In the *adaptive* method, a population index ξ which can measure the diversity of the population is considered as the feedback information to direct the change of the values of pc_{ls} and nm_{ls}.

Let the normalized Hamming distance between two individuals $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})$ and $\mathbf{x}_j = (x_{j1}, \ldots, x_{jn})$ be defined by:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{n} |x_{ik} - x_{jk}|}{n} \tag{6.1}$$

and ξ is calculated by the following formula:

$$\xi = \frac{\sum_{i=1}^{pop\_size} d(\mathbf{x}^*, \mathbf{x}_i)}{pop\_size}, \tag{6.2}$$

where $\mathbf{x}^*$ denotes the best individual achieved so far. Obviously, the index ξ can measure the convergence state of the population via the Hamming distance. When ξ decreases to zero, it means that the population has lost its diversity absolutely.

With the definition of ξ, pc_{ls} and nm_{ls} can be calculated as follows:

$$pc_{ls} = min\{\alpha \cdot \xi \cdot (pc_{ls}^{max} - pc_{ls}^{min}) + pc_{ls}^{min}, pc_{ls}^{max}\} \tag{6.3}$$

$$nm_{ls} = min\{\beta \cdot \xi \cdot (nm_{ls}^{max} - nm_{ls}^{min}) + nm_{ls}^{min}, nm_{ls}^{max}\}, \tag{6.4}$$

where pc_{ls}^{max} and pc_{ls}^{min} are the preset maximum and minimum value of pc_{ls} respectively (pc_{ls}^{max}=0.6 and pc_{ls}^{min}=0.1 in the later experiments), nm_{ls}^{max} and nm_{ls}^{min} are the preset maximum and minimum value of nm_{ls} respectively (nm_{ls}^{max}=4 and nm_{ls}^{min}=1 in the later experiments), and α and β are the predefined constants to control the decreasing or increasing speed of pc_{ls} and nm_{ls} respectively. From these formulae, it is easy to understand that both GCHC and SMHC exhibit a wide range LS operations in the presence of a high population diversity (i.e., when $\xi \rightarrow 1$) as a result of $pc_{ls} \rightarrow pc_{ls}^{max}$ and $nm_{ls} \rightarrow nm_{ls}^{max}$. This may help algorithms find the optimum (maybe local optimum) more quickly. However, when the population is converging (i.e., when $\xi \rightarrow 0$), $pc_{ls} \rightarrow pc_{ls}^{min}$ and $nm_{ls} \rightarrow nm_{ls}^{min}$, which limits the LS operations in a very small range in order to perform more efficient local improvement for the selected individual *chr*.

6.2.3 Adaptive Learning Mechanism in Multiple LS Operators

It has been reported that multiple LS operators can be employed in an MA framework [22, 33, 34]. This is because each LS operator makes a biased search, which makes a method efficient for some classes of problems but not efficient for others. That is, LS is problem-dependent. Therefore, how to achieve improved LS operators and avoid utilizing inappropriate LS methods becomes a very important issue. In order to address this problem, many researchers have used multiple LS methods in their MAs. In comparison with traditional MAs that use a single LS operator throughout the run, MAs with multiple LS methods can usually obtain a better performance.

The key idea of using multiple LS operators in MAs is to promote the cooperation and competition of different LS operators, enabling them to work together to accomplish the shared optimization goal. Some researchers [16, 26] have suggested that multiple LS operators should be executed simultaneously on those individuals that are selected for local improvements and that a certain learning mechanism should be adopted to give the efficient LS methods greater chances to be chosen in the later stage. However, Neri *et al.* [23] have also proposed a multiple LS based MA with a non-competitive scheme, where different LS methods can be activated during different population evolution periods. Inspired by these researches, a learning mechanism is discussed to hybridizes the GCHC and SMHC methods described in Section 6.2.2 effectively and an adaptive hill climbing (AHC) strategy is introduced in this study.

In AHC, the GCHC and SMHC operators are both allowed to work in the whole LS loop and are selected by probability to execute one step LS operation at every generation when the MA is running. Let p_{gchc} and p_{smhc} denote the probabilities of applying GCHC and SMHC to the individual that is used for a local search respectively, where p_{gchc}+p_{smhc}=1. At the start of this strategy, p_{gchc} and p_{smhc} are both set to 0.5, which means giving a fair competition chance to each LS operator. As each LS operator always makes a biased search, the LS operator which produces more improvements should be given a greater selection probability. Here, an adaptive learning approach is used to adjust the value of p_{gchc} and p_{smhc} for each LS operator. Let η denote the improvement degree of the selected individual when one LS operator is used to refine it and η can be calculated by:

$$\eta = \frac{|f_{imp} - f_{ini}|}{f_{ini}}, \tag{6.5}$$

where f_{imp} is the final fitness of the selected individual *chr* after applying LS and f_{ini} is its initial fitness before executing LS operation. At each generation, the degree of improvement of each LS operator is calculated when a predefined number (num_{ls}) of iterations is achieved and then p_{gchc} and p_{smhc} are re-calculated to proceed with the next LS operation.

Suppose $\eta_{gchc}(t)$ and $\eta_{smhc}(t)$ respectively denote the total improvement of GCHC and SMHC at iteration t. The LS selection probabilities $p_{gchc}(t+1)$ and $p_{smhc}(t+1)$ at iteration $(t+1)$ can be calculated orderly by the following formulae:

$$p_{gchc}(t+1) = p_{gchc}(t) + \Delta \cdot \eta_{gchc}(t), \tag{6.6}$$

$$p_{smhc}(t+1) = p_{smhc}(t) + \Delta \cdot \eta_{smhc}(t), \tag{6.7}$$

$$p_{gchc}(t+1) = \frac{p_{gchc}(t+1)}{p_{gchc}(t+1) + p_{smhc}(t+1)}, \tag{6.8}$$

$$p_{smhc}(t+1) = 1 - p_{gchc}(t+1), \tag{6.9}$$

where Δ signifies the relative influence of the degree of the improvement on the selection probability. The AHC operator can be expressed by the pseudo-code in Fig. 6.4.

From the above discussion, the two different HC strategies, GCHC and SMHC, may not only cooperate to improve the quality of individuals, but also compete with each other to achieve a greater selection probability in the running process of AHC. To promote competition between them, the selection probability of LS operators can be re-calculated according to an adaptive learning mechanism where the LS operator with a higher fitness improvement is rewarded with more chance of being chosen for the subsequent individual refinement.

```
Procedure AHC($chr$):
begin
  if $p_{gchc}$ and $p_{smhc}$ are not initialized then
    set $p_{gchc} := p_{smhc} := 0.5$;
  calculate($\xi, pc_{ls}, nm_{ls}$);
  set $\eta_{gchc} = \eta_{smhc} := 0$;
  for $i := 1$ to $num_{ls}$ do
    if $random() < p_{gchc}$ then // GCHC is selected
      GCHC($chr$);
      update($\eta_{gchc}$);
    else // SMHC is selected
      SMHC($chr$);
      update($\eta_{smhc}$);
  endfor
  recalculate($p_{gchc}, p_{smhc}$);
end;

AHC's denotations are the same as those for GCHC and SMHC
```

Fig. 6.4 Pseudo-code for the AHC operator

6.2.4 Diversity Maintaining

For stationary optimization problems, where the fitness landscape or objective function does not change during the course of computation, LS operators in MAs are designed for exploiting information in the current population and genetic operators, for example, mutation, are mostly responsible for enhancing the diversity of population in order to make an efficient jump from a local optimum. Generally speaking, the population will converge to a small area in the whole search space as a result of keeping the sufficient exploitation for the global optimum. Therefore, MAs may gradually loose their population diversity during the running. However, in dynamic environments, the fitness landscape may change over time. That is, the current optimum point may become a local optimum and the past local optimum may become a new global optimum point. Considering that a spread-out population can adapt to these changes more easily, it is very important and necessary to maintain sufficient diversity of the population for MAs all the time.

The immigrants scheme is a quite simple and common method to maintain the diversity level of the population through substituting a portion of individuals in the population with the same number of newly-generated immigrants every generation. Obviously, the method of generating new immigrants becomes a very important issue in this strategy. In the original scheme, immigrants are designed to be generated randomly. However, the random immigrants introduced may divert the searching force of an algorithm and hence degrade the performance due to their lower fitness level. Another thing to notice is that random individuals are not helpful for

```
Procedure GenerateImmigrants():
begin
  for $i := 1$ to $num_{im}$ do
    $P_{im}[i]$ := mutation($elite$, $pm_{im}$);
    evaluate($P_{im}[i]$);
  endfor
  replace the worst $num\_im$ individuals in the
    current population with $P_{im}$
end

Denotations:
  $pm_{im}$: the mutation probability in generating immigrants
  $num\_im$: the number of generated immigrants
  $elite$: the best fitness individual in the population
  $P_{im}$: the generated immigrants set
```

Fig. 6.5 Pseudo-code for the general immigrants scheme

improving the diversity level when being inserted into a spread-out population. In the literature [44], a special immigrants scheme, called elitism-based immigrants, is proposed. In this approach, the elitism individual, which is the best fitness individual in the population, is used as a base to generate a set of immigrant individuals iteratively by a simple mutation. The key idea behind this scheme is that the immigrants can be guided to make a biased detection around the elite. However, it is also noticeable that this bias may take no effect in a converging population since all the individuals in the current population have distributed around its elite.

Inspired by the complementarity mechanism in nature, a primal-dual mapping operator, where two chromosomes with the maximal distance in the solution space are defined as a pair of primal-dual chromosomes, was be proposed and applied for GAs in dynamic environments. Here, a new immigrants scheme, called dual-based immigrants, is proposed with the hybridization the primal-dual mechanism and elitism-based immigrants scheme. The dual-based immigrants are generated by executing a simple mutation to the dual of the elite individual and replace the same number of worst individuals in the current population. It is obvious that the hyper-immigrants scheme can enhance the diversity level with the maximal degree, which is helpful to improve the performance of EAs with a converging population in dynamic environments especially in the significantly changing environments.

It seems very interesting that the above-mentioned three different immigrants schemes, that is, elitism-based immigrants, random immigrants, and dual-based immigrants, can actually be described in a general framework. As shown in Fig. 6.5, all immigrants are considered to be generated from mutating the elite individual with a probability. It is easy to understand that the mutation probability pm_{im} can be used to control the categories of immigrants. The generated immigrants belong to the elitism-based immigrants when the value of pm_{im} is very small (e.g., $pm_{im} \to 0$), while the dual-based immigrants can be achieved when executing the mutation to $elite$ with a large probability (e.g., $pm_{im} \rightarrow 1$). When $pm_{im} = 0.5$, the mutation operation always creates random immigrants in fact.

The following problem to be addressed is how to set the mutation probability pm_{im}. Here, we will utilize the index of population diversity ξ, discussed in Section 6.2.2, to calculate the value of pm_{im} by the following formula.

$$pm_{im} = max\{min\{(pm_{im}^{max} + pm_{im}^{min} * random() - \xi), pm_{im}^{max}\}, pm_{im}^{min}\} \qquad (6.10)$$

where $random()$ is a random number between 0 and 1, pm_{im}^{max} and pm_{im}^{min} are the preset maximum and minimum value of pm_{im}, respectively ($pm_{im}^{max} = 0.95$ and $pm_{im}^{min} = 0.05$ in the later experiments). From this formula, it is easy to see that the elitism-based immigrants are inserted into the population when the population has a low diversity level ($\xi \rightarrow 1$) due to pm_{im} approaches to pm_{im}^{min}, while the dual-based immigrants are generated when the population is converging ($\xi \rightarrow 0$) as a result of $pm_{im} \rightarrow pm_{im}^{max}$.

6.2.5 Balance between Local Search and Diversity Maintaining

Based on the above description, an adaptive hill climbing operator, where two different LS operators are used in a cooperation fashion, is proposed to enhance the exploitation capacity of algorithm and hence accelerate its tracking the optimum, while a hybrid immigrants scheme, where the different category of immigrants are generated according to the convergence status of population, is used to maintain the population diversity of algorithm and then improve its performance in dynamic environments. However, it is noticeable that the extra computation cost, which means the numbers of fitness evaluation in this paper, would be produced during executing LS and generating immigrants ($step\_ls$ in LS and num_im in diversity scheme) every generation.

One main question that follows when both LS operation and diversity maintaining strategy are introduced into an algorithm is how to keep the balance between them via the extra computational cost. As the generation index is used to set the change period of environment in the later experiments, it is necessary to maintain a constant number of evaluations in each generation in order to have fair comparisons among our investigated MAs and other peer EAs. Therefore, the additional number of fitness evaluations per generation, denoted as num_aepg, is also fixed, that is, $num_{ls} + num_{im} = num\_aepg$. Based on this formula, two different methods can be considered to calculate the values of num_{ls} and num_{ls}.

The first one is dominated by the effect of LS that the value of num_{ls} is firstly calculated and then num_{im}'s value is achieved by $(num\_aepg - num_{ls})$. Let $num_{vls}(t)$ denote the number of valid LS operations, which means the corresponding index $\eta > 0$ after executing one LS operation, at the t-th generation, $num_{ls}(t+1)$ can be calculated by:

$$num_{ls}(t+1) = \begin{cases} max\{num_{ls}(t) \cdot \lambda_0, num_{ls}^{min}\}, \text{ if } num_{vls}(t) > 0 \\ min\{num_{ls}(t)/\lambda_0, num_{ls}^{max}\}, \text{ else} \end{cases} \tag{6.11}$$

where λ_0 is a preset constant between 0 and 1, num_{ls}^{max} and num_{ls}^{min} are the preset maximum and minimum value of num_{ls} respectively ($num_{ls}^{max} = 0.75 * num\_aepg$ and $num_{ls}^{min} = 0.25 * num\_aepg$ in the later experiments). It is easy to understand from this formula that the LS operation would be further enhanced at the next generation, that is, the value of num_{ls} is increased, once there exists even one valid LS operation at a step.

The second method is dominated by the diversity level of the population, i.e., the value of num_{im} is firstly calculated and then num_{ls}'s value is achieved by $(num\_aepg - num_{im})$. Let $num_{im}(t)$ denote the number of generated immigrants at the t-th generation, $num_{ls}(t+1)$'s value can be calculated by the following formula.

$$num_{im}(t+1) = \begin{cases} max\{num_{im}(t) \cdot \lambda_1, num_{im}^{min}\}, \text{ if } \xi > \xi_0 \\ min\{num_{im}(t)/\lambda_1, num_{im}^{max}\}, \text{ else} \end{cases} \tag{6.12}$$

where λ_1 is a preset constant between 0 and 1 to signify the influence degree of the population diversity upon num_{im}, ξ_0 is also a preset constant between 0 and 1, num_{im}^{max} and num_{im}^{min} are the preset maximum and minimum value of num_{im} respectively ($num_{im}^{max} = 0.75 * num\_aepg$ and $num_{im}^{min} = 0.25 * num\_aepg$ in the later experiments). It is easy to see from this formula that the number of generated immigrants would be increased to maintain the diversity level of the population once its converge degree is below a threshold (ξ_0).

Therefore, the framework of our proposed MAs in this chapter, which hybridizes the AHC operator and immigrants scheme for DOPs, can be summarized by the pseudo-code in Fig. 6.6.

6.3 Dynamic Test Environments

In this chapter, a series of dynamic test environments are constructed by a specific dynamic problem generator from a set of well-studied stationary problems.

Four 100-bit binary-coded functions, denoted OneMax, Plateau, RoyalRoad, and Deceptive respectively, are selected as the stationary functions to construct dynamic test environments. Each stationary function consists of 25 copies of 4-bit building blocks and has an optimum value of 100. Each building block for the four functions is a unitation-based function, as shown in Fig. 6.7. The unitation function of a bit string returns the number of ones inside the string. The building block for OneMax is an OneMax subfunction, which aims to maximize the number of ones in a bit string. The building block for Plateau contributes 4 (or 2) to the total fitness if its unitation is 4 (or 3); otherwise, it contributes 0. The building block for RoyalRoad contributes 4 to the total fitness if all its four bits are set to one; otherwise, it contributes 0. The building block for Deceptive is a fully deceptive sub-function. Generally speaking, the four functions have an increasing difficulty for GAs in the order from OneMax to Plateau, RoyalRoad to Deceptive.

```
Procedure Proposed GA-based MA:
begin
  parameterize();
  $t := 0$;
  initializePopulation($P(0)$);
  evaluatePopulation($P(0)$);
  calculate($\xi, pc_{ls}, nm_{ls}, pm_{im}$);
  set $num_{ls}(0) := num_{im}(0) := num\_aepg/2$;
  $elite$ := selectForLocalSearch($P(0)$);
  AHC($elite$);
  GenerateImmigrants();
  repeat
    $P'(t)$ := selectForReproduction($P(t)$);
    $P''(t)$ := crossover($P'(t)$);
    mutate($P''(t)$);
    evaluatePopulation($P''(t)$);
    $P(t+1)$ := selectForSurvival($P''(t)$, $P(t)$);
    calculate($\xi, pc_{ls}, nm_{ls}, pm_{im}$);
    calculate($num_{ls}(t+1), num_{im}(t+1)$);
    $elite$ := selectForLocalSearch($P(t+1)$);
    AHC($elite$);
    GenerateImmigrants();
    $t := t+1$;
  until a stop condition is met
end;
```

Fig. 6.6 Pseudo-code for the proposed GA-based MAs

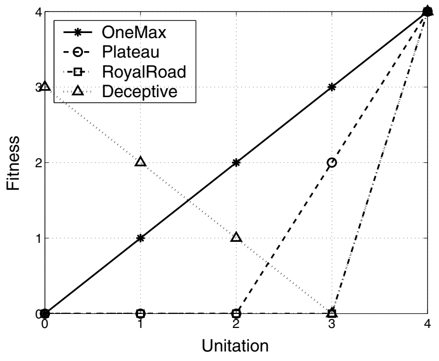

Fig. 6.7 The building blocks for the four stationary functions selected to construct dynamic test problems in this chapter

In [42, 47], an XOR DOP generator was proposed. The XOR DOP generator can generate dynamic environments from any binary-encoded stationary function $f(\mathbf{x})$ ($\mathbf{x} \in \{0,1\}^l$) by a bitwise exclusive-or (XOR) operator. The environment is changed every τ generations. For each environmental period k, an XOR mask $\mathbf{M}(k)$ is incrementally generated as follows:

$$\mathbf{M}(k) = \mathbf{M}(k-1) \oplus \mathbf{T}(k), \tag{6.13}$$

where "$\oplus$" is the XOR operator (i.e., $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$) and $\mathbf{T}(k)$ is an intermediate binary template randomly created with $\rho \times l$ ones for the k-th environmental period. For the first period $k = 1$, $\mathbf{M}(1) = \mathbf{0}$. Then, the population at generation t is evaluated as:

$$f(\mathbf{x},t) = f(\mathbf{x} \oplus \mathbf{M}(k)), \tag{6.14}$$

where $k = \lceil t/\tau \rceil$ is the environmental index. One advantage of this XOR generator lies in that the speed and severity of environmental changes can be easily tuned. The parameter τ controls the speed of changes while $\rho \in (0.0, 1.0)$ controls the severity of changes. A bigger ρ means more severe changes while a smaller τ means more frequent changes.

The dynamic test environments used in this paper are constructed from the four stationary functions using the aforementioned XOR DOP generator. The change severity ρ parameter is set to 0.1, 0.2, 0.5, and 0.9 respectively in order to examine the performance of algorithms in dynamic environments with different severities: from slight change ($\rho = 0.1$ or 0.2) to moderate variation ($\rho = 0.5$) to intense change ($\rho = 0.9$). The change speed parameter τ is set to 10, 50, and 100 respectively, which means that the environment changes very fast, in the moderate speed, and slowly respectively.

In total, a series of 12 different dynamic problems are constructed from each stationary test problem. The dynamics parameter settings are summarized in Table 6.1.

Table 6.1 The index table for dynamic parameter settings

τ	Environmental Dynamics Index			
10	1	2	3	4
50	5	6	7	8
100	9	10	11	12
$\rho \rightarrow$	0.1	0.2	0.5	0.9

6.4 Experimental Study

6.4.1 Experimental Design

In this section, experiments are carried out in order to study the major features of our proposed MAs and to compare their performance with several existing peer algorithms where similar dualism and immigrants methods are also used. The following abbreviations represent the algorithms considered in this paper:

- SGA: Standard GA;
- SGAr: SGA with restart from scratch whenever the environment changes;
- EIGA: GA with the elitism-based immigrants scheme [44];
- RIGA: GA with the random immigrants scheme;
- DIGA: GA with the dual-based immigrants scheme (seen in Section 6.2.4);
- CHMA: MA with the GCHC operator;
- MHMA: MA with the SMHC operator;
- AHMA: MA with the AHC operator;
- EIAHMA: AHMA with the elitism-based immigrants scheme;
- RIAHMA: AHMA with the random immigrants scheme;
- DIAHMA: AHMA with the dual-based immigrants scheme;
- IMAHMA: AHMA with the proposed immigrants scheme in Section 6.2.4;

Some parameters in all algorithms were set as follows. The total number of evaluations per generation was always set to 120 for all algorithms, which means the population size (*pop_size*) is equal to 120 for SGA and SGAr. In all MAs, EIGA, RIGA and DIGA, *pop_size* was set to 100 and the additional number of fitness evaluations per generation (*num_aepg*) was set to 20. The uniform crossover probability *pc* was set to 0.6 and the bit-wise mutation probability *pm* was set to 0.01 for all GAs and MAs. The specific parameters in our MAs were set as follows: $\alpha = \beta = 1$, $\Delta = 4$, $\lambda_0 = \lambda_1 = 0.9$, and $\theta_0 = 0.5$. Other parameters in the studied peer algorithms were set the same as their original settings.

For each experiment of an algorithm on a test problem, 20 independent runs were executed with the same set of random seeds. For each run of an algorithm on a DOP, 10 environmental changes were allowed and the best-of-generation fitness was recorded per generation. The overall offline performance of an algorithm is defined as the best-of-generation fitness averaged across the number of total runs and then averaged over the data gathering period, as formulated below:

$$\overline{F}_{BG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BG_{ij}}\right), \tag{6.15}$$

where G is the number of generations (i.e., $G = 10 * \tau$), $N = 20$ is the total number of runs, and $F_{BG_{ij}}$ is the best-of-generation fitness of generation i of run j.

In order to measure the behavior of an algorithm during the course of running, another numeric measure is defined as the best-of-generation fitness averaged across the number of total runs and then averaged from the last change generation τ' to the current generation t. More formally, the running offline performance is defined as:

$$\overline{F}_{BG_t} = \frac{1}{t-\tau'}\sum_{i=\tau'}^{t-\tau'}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BG_{ij}}\right) \tag{6.16}$$

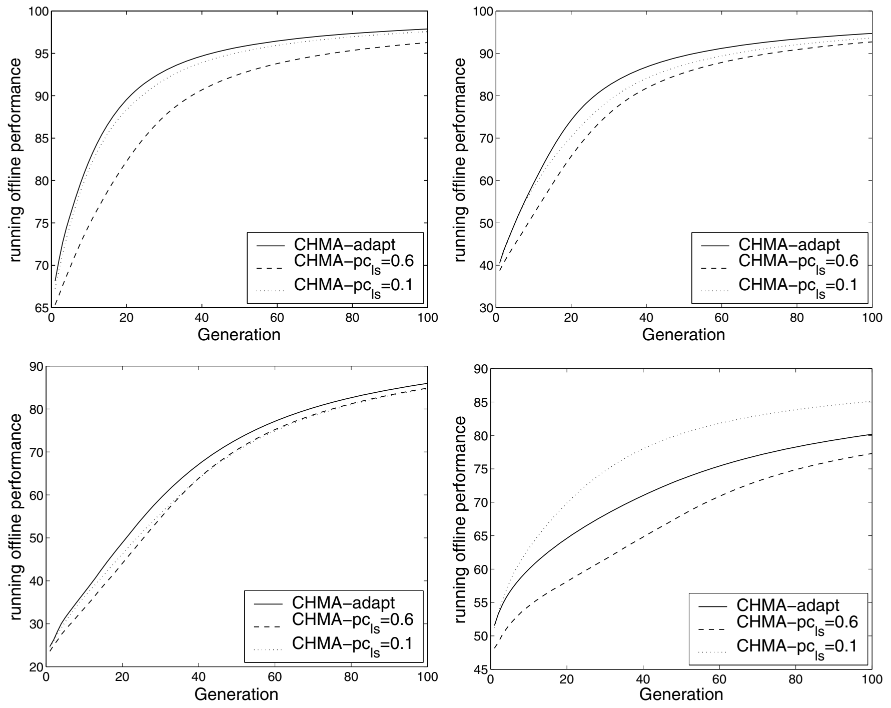

Fig. 6.8 Experimental results with respect to the running offline performance of CHMAs with different pc_{ls} settings on stationary test problems: (a) OneMax, (b) Plateau, (c) RoyalRoad, and (d) Deceptive

6.4.2 Experimental Study on the Effect of LS Operators

In the experimental study on LS operators, we first study the influence of different settings of pc_{ls} in CHMA and nm_{ls} in MHMA, with the aim of determining a robust setting for these two parameters. In particular, we have implemented CHMA just on stationary test problems. Three different settings for pc_{ls} were used: $pc_{ls} = 0.6$ and $pc_{ls} = 0.1$ in the *deterministic* setting and $pc_{ls}^{max} = 0.6$ and $pc_{ls}^{min} = 0.1$ in the *adaptive* setting scheme (see Section 6.2.2). For each run of an algorithm on each problem, the maximum allowable number of generations was set to 100[1]. The experimental results are shown in Fig. 6.8, where the data were averaged over 20 runs. The results on the Plateau problem are similar to the results on the RoyalRoad problem and are not shown in Fig. 6.8.

From Fig. 6.8, it can be seen that CHMA with *adaptive* pc_{ls} always outperforms CHMAs with the *deterministic* value of pc_{ls} on the OneMax, Plateau and RoyalRoad problems and that a smaller pc_{ls} can help CHMA obtain a better perfor-

[1] The number of maximum allowable fitness evaluations is actually 12000 since each algorithm has 120 fitness evaluations per generation.

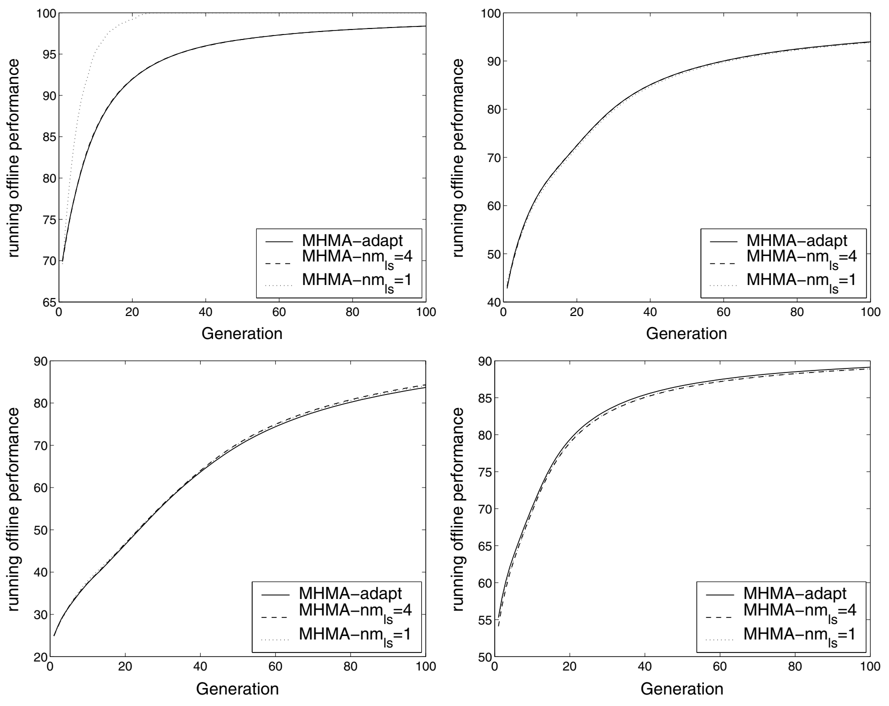

Fig. 6.9 Experimental results with respect to the running offline performance of MHMAs with different nm_{ls} settings on stationary test problems: (a) OneMax, (b) Plateau, (c) RoyalRoad, and (d) Deceptive

mance on the Deceptive problem. Hence, the *adaptive* setting scheme for pc_{ls} will always be used in the following experiments considering that the *deterministic* setting scheme is problem-dependent and the *adaptive* scheme for pc_{ls} always shows a better adaptive capacity on different problems.

Similar experiments were also carried out to test the influence of different settings of nm_{ls} on the performance of MHMA. The value of nm_{ls} was set to 4 and 1 respectively for the *deterministic* scheme and $nm_{ls}^{max} = 4$ and $nm_{ls}^{min} = 1$ in the *adaptive* setting scheme (see Section 6.2.2). The experimental results with respect to the running offline performance are presented in Fig. 6.9.

From Fig. 6.9, it can be observed that the performance curves of the three MHMAs almost overlap together on the Plateau, RoyalRoad and Deceptive problems except that MHMA with $nm_{ls} = 1$ performs better than MHMA with adaptive nm_{ls} and MHMA with $nm_{ls} = 4$ on the OneMax problem. This indicates that adaptively varying the search range of the SMHC operator may not improve the performance of MHMA remarkably. Hence, the value of nm_{ls} will always be set to 1 in the later experiments.

In the following experiments, we investigate the performance of AHMA, MHMA, CHMA and SGA on the stationary test problems in order to examine the validity of

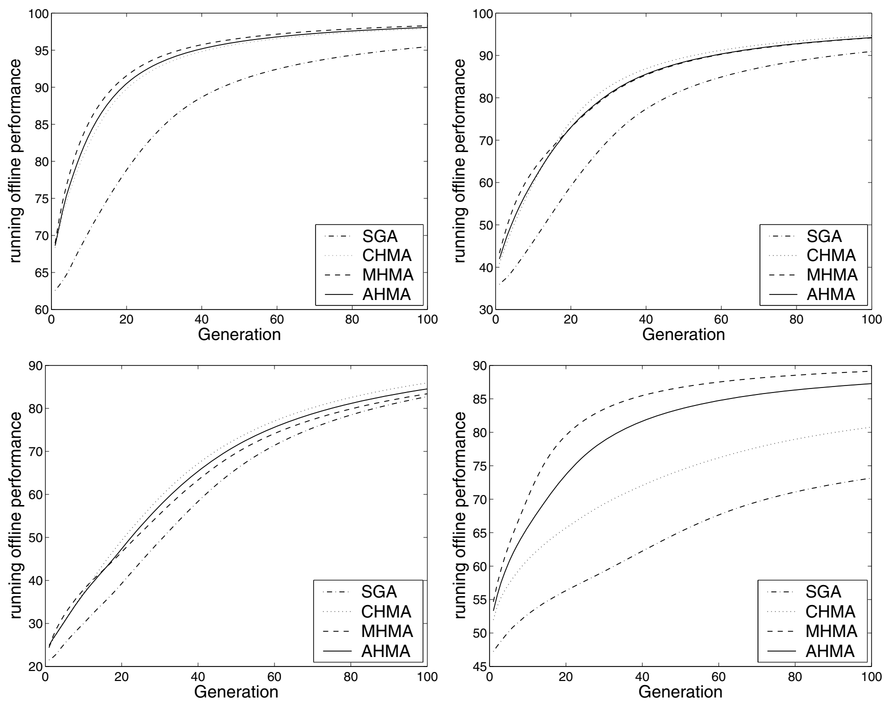

Fig. 6.10 Experimental results with respect to the running offline performance of MAs and SGA on stationary test problems: (a) OneMax, (b) Plateau, (c) RoyalRoad, and (d) Deceptive

our proposed AHC operator. The experimental results with respect to the running offline performance are shown in Fig. 6.10.

From Fig. 6.10, it can be seen that all MAs outperform SGA significantly on all test problems. This shows that proper LS techniques (here AHC in AHMA, SMHC in MHMA, and GCHC in CHMA) can help MAs obtain a much better performance than SGA. Of course, these conclusions have been drawn by many researchers. MHMA exhibits the best performance on the OneMax and Deceptive problems, while CHMA performs best on the Plateau and RoyalRoad problems among the three MAs, which shows that the effect of LS operators is problem-dependent. It can also be seen that AHMA always shows good adaptivity on the four test problems, where AHMA performs better than CHMA on the OneMax and Deceptive problems and better than MHMA on the Plateau and RoyalRoad problems. The results indicate that AHC always does well although it needs to take some time to adjust its LS strategy. Since it is almost impossible for an algorithm to achieve all the characters of a problem in advance, the combination of multiple LS operators within a single MA framework is a good choice for solving optimization problems.

In the above experimental studies, only the elite chromosome is selected for local refinement in order to decrease extra cost due to the fitness evaluations by LS

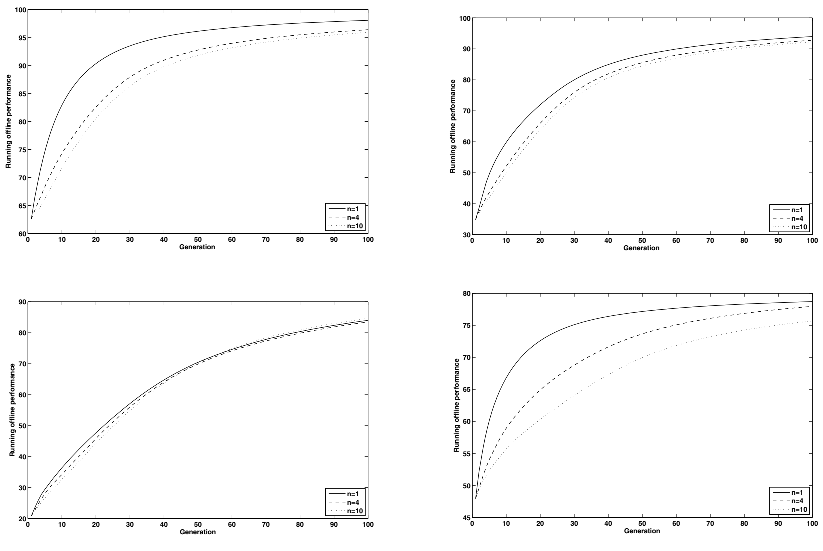

Fig. 6.11 Experimental results with respect to the running offline performance of AHMAs on stationary test problems: (a) OneMax, (b) Plateau, (c) RoyalRoad, and (d) Deceptive

operations. In the final experiments on LS operators, we further investigate the influence of the number of selected individuals for LS upon the performance of AHMA. In order to make a fair comparison, the number of additional fitness evaluations per generation (*num_aepg*) is always set to 20 and the number of selected individuals (n) is set to 1, 4 and 10, respectively, which means the corresponding iteration number of LS is 20, 5 and 2, respectively. In addition, only the best n individuals in the population would undergo the AHC operation. The experimental results with respect to the running offline performance are shown in Fig. 6.11.

From Fig. 6.11, it can be seen that the performance of AHMAs degrades with the increment of the number of selected individuals for LS, especially on the OneMax, RoyalRoad and Deceptive problems. This means that only executing sufficient local refinement upon the best individual *elite* is a good choice when the extra cost is limited.

6.4.3 Experimental Study on the Effect of Diversity Maintaining Schemes

Immigrants scheme is a common strategy to address the convergence problem of EAs in dynamic environments. In Section 6.2.4, three different immigrants schemes are integrated into a general framework, where different categories of immigrants can be generated by executing the simple mutation on the best fitness individual (*elite*) with different probabilities. In order to investigate the effect of proposed immigrants schemes upon the performance of algorithms in dynamic environments, we firstly carry out the following experiments on AHMAs with different immigrants schemes on DOPs with τ=50 and ρ set to 0.1, 0.2, 0.5 and 0.9, respectively. In all AHMAs, the first method of calculating the values of num_{ls} and num_{im} are used.

The experimental results with respect to the overall offline performance are presented in Table 6.2. The corresponding statistical results of comparing algorithms by the one-tailed t-test with 38 degrees of freedom at a 0.05 level of significance are given in Table 6.3. In Table 6.3, the t-test results regarding Alg. 1$-$Alg. 2 are shown as "$+$", "$-$", or "$\sim$" when Alg. 1 is significantly better than, significantly worse than, or statistically equivalent to Alg. 2, respectively. From Table 6.2 and Table 6.3, several results can be observed and are analyzed below.

First, EIAHMA performs better than RIAHMA and DIAHMA on all dynamic OneMax problems and dynamic Plateau problems when the change severity is small. This is because the immigrant individuals generated by the elitism mechanism can always make a positive guide for the search of the algorithm on the OnmeMax problem. In addition, the similar effect of this immigrants scheme can be obtained when the environmental change is slight, e.g., dynamic Plateau problems with $\rho = 0.1$ and $\rho = 0.2$. It can also be seen that the performance of EIAHMA begins to degrade with the increasing of the value of ρ. When $\rho = 0.5$, EIAHMA performs much worse than RIAHMA on dynamic Plateau, RoyalRoad and Deceptive problems. When the value of ρ increases to 0.9, EIAHMA is always beaten by DIAHMA with a great degree.

Second, RIAHMA exhibits the best performance on most dynamic environments when a random environmental change occurs (ρ=0.5). This is easy to understand. When the environment changes with $\rho = 0.5$, almost all building blocks found so far are demolished. Obviously, RIAHMA can adapt to this environmental change more easily since the newly-generated random immigrants always do better than those biased immigrants, that is, the elitism-based immigrants in EIAHMA and the dual-based immigrants in DIAHMA, in re-achieving the demolished building blocks.

Third, DIAHMA performs better than EIAHMA and RIAHMA on dynamic RoyalRoad problems when $\rho = 0.9$ and on all dynamic Deceptive problems. The reason lies in that the dual mechanism may help DIAHMA react to significant environmental changes rapidly and also enable it to escape from the deceptive attractor in the Deceptive problem.

Fourth, IMAHMA always exhibits good performance in most dynamic environments except that it is beaten entirely by EIAHMA on dynamic OneMax problems with the different settings of change severity ρ. In IMAHMA, the probability of generating immigrants (pm_{im}) can be calculated adaptively according to the convergence degree of population. It means IMAHMA can generate different categories of immigrants to adapt well to dynamic environments with different change severities. The experimental results of the good adaptivity of IMAHMA in dynamic environments confirm our expectation of introducing the proposed immigrants scheme in Section 6.2.4 to AHMA for DOPs.

Table 6.2 Experimental results with respect to overall offline performance of AHMAs on dynamic test problems

Dynamics		OneMax Problem			
τ	ρ	IMAHMA	EIAHMA	RIAHMA	DIAHMA
50	0.1	97.11±0.27	97.58±0.19	97.00±0.27	97.08±0.28
50	0.2	95.11±0.37	95.86±0.27	95.03±0.27	95.10±0.24
50	0.5	92.99±0.34	93.54±0.39	93.73±0.32	92.97±0.30
50	0.9	97.18±0.17	97.52±0.31	97.06±0.19	97.09±0.21
Dynamics		Plateau Problem			
τ	ρ	IMAHMA	EIAHMA	RIAHMA	DIAHMA
50	0.1	93.62±0.45	93.76±0.82	93.62±0.42	93.48±0.54
50	0.2	87.53±0.71	87.93±0.98	87.41±0.93	87.41±0.83
50	0.5	81.94±1.00	79.51±1.70	83.65±0.77	81.25±1.11
50	0.9	93.60±0.66	93.48±1.95	93.42±0.60	93.54±0.46
Dynamics		RoyalRoad Problem			
τ	ρ	IMAHMA	EIAHMA	RIAHMA	DIAHMA
50	0.1	85.77±1.19	80.97±3.15	85.64±1.14	85.76±1.13
50	0.2	73.96±1.52	69.76±1.81	73.70±1.82	73.88±1.82
50	0.5	63.83±1.83	55.08±2.20	65.65±1.73	61.55±1.36
50	0.9	85.23±1.02	79.67±4.93	84.93±1.21	85.37±0.94
Dynamics		Deceptive Function			
τ	ρ	IMAHMA	EIAHMA	RIAHMA	DIAHMA
50	0.1	83.44±0.46	76.57±1.42	76.72±1.66	83.99±0.72
50	0.2	81.24±0.47	74.47±0.82	74.14±1.02	82.19±0.54
50	0.5	80.29±0.51	73.50±0.72	74.34±0.62	81.19±0.70
50	0.9	84.07±0.60	77.02±1.45	77.22±1.71	84.06±0.60

Table 6.3 The *t*-test results of comparing the overall offline performance of AHMAs on dynamic test problems

t-test Result	OneMax				Plateau				RoyalRoad				Deceptive			
$\tau = 50, \rho \Rightarrow$	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9
IMAHMA − EIAHMA	−	−	−	−	∼	∼	+	∼	+	+	+	+	+	+	+	+
IMAHMA − RIAHMA	∼	∼	−	+	∼	∼	−	∼	∼	∼	−	∼	+	+	+	+
IMAHMA − DIAHMA	∼	∼	∼	+	∼	∼	+	∼	∼	∼	+	∼	−	−	−	∼
EIAHMA − RIAHMA	+	+	∼	+	∼	+	−	∼	−	−	−	−	∼	∼	−	∼
EIAHMA − DIAHMA	+	+	+	+	∼	+	−	∼	−	−	−	−	−	−	−	−
RIAHMA − DIAHMA	∼	∼	+	∼	∼	∼	+	∼	∼	∼	+	∼	−	−	−	−

Both LS and diversity maintaining scheme require to cost a certain number of additional fitness evaluations. Therefore, it becomes a key problem to balance the two operations considering that the total additional number of evaluations per generation (*num_aepg*) is always constant. In Section 6.2.5, two different schemes are proposed to calculate the corresponding values of additional evaluations consumed

Table 6.4 Experimental results with respect to overall offline performance of IMAHMAs on dynamic test problems

Dynamics		OneMax Problem		
τ	ρ	IMAHMA0	IMAHMA1	IMAHMA2
50	0.1	96.73±0.20	97.13±0.24	96.29±0.24
50	0.2	93.86±0.38	94.86±0.29	92.66±0.45
50	0.5	91.45±0.39	92.89±0.30	90.04±0.52
50	0.9	96.66±0.20	97.04±0.20	96.16±0.24
Dynamics		Plateau Problem		
τ	ρ	IMAHMA0	IMAHMA1	IMAHMA2
50	0.1	92.78±0.62	93.48±0.63	92.39±0.60
50	0.2	86.66±0.86	87.70±0.77	85.44±0.94
50	0.5	80.54±0.96	81.43±1.00	79.55±0.86
50	0.9	92.79±0.54	93.45±0.55	92.15±0.57
Dynamics		RoyalRoad Problem		
τ	ρ	IMAHMA0	IMAHMA1	IMAHMA2
50	0.1	84.99±1.31	85.18±1.76	84.72±1.48
50	0.2	73.74±1.58	74.04±1.62	73.17±1.60
50	0.5	63.71±2.07	64.81±2.14	64.42±1.79
50	0.9	85.14±1.55	85.95±1.34	85.30±1.25
Dynamics		Deceptive Problem		
τ	ρ	IMAHMA0	IMAHMA1	IMAHMA2
50	0.1	83.25±0.57	83.62±0.53	81.26±0.92
50	0.2	79.69±0.82	81.39±0.48	76.51±0.93
50	0.5	77.74±0.61	79.99±0.46	73.98±0.84
50	0.9	82.77±0.87	83.59±0.72	80.88±1.19

by LS (num_{ls}) and immigrants scheme (num_{im}), respectively. In the following experiments, we will examine the effect of proposed schemes upon the performance of IMAHMAs in dynamic environments. For the sake of convenient description of the experiments, IMAHMA0, IMAHMA1 and IMAHMA2 are used to denote IMAHMA with a *deterministic* scheme where both num_{ls} and num_{im} are fixed to $num\_aepg/2$, IMAHMA with the first dominated scheme, and IMAHMA with the second dominated scheme, respectively.

The experimental results respect to the overall offline performance are presented in Table 6.4 and the corresponding statistical results of comparing algorithms by the one-tailed t-test with 38 degrees of freedom at a 0.05 level of significance are given in Table 6.5. From Table 6.4 and Table 6.5, several results can be observed and are analyzed below.

First, IMAHMA1 performs better than IMAHMA0 and IMAHMA2 on all test problems. In IMAHMA1, the value of additional fitness evaluations costed by LS (num_{ls}) is firstly calculated based on the effect of LS. When LS helps to improve the quality of individuals, that is, the number of valid LS (num_{vls}) is larger than zero, num_{ls}'s value would increase in order to execute more sufficient local refinement,

Table 6.5 The t-test results of comparing the overall offline performance of IMAHMAs on dynamic test problems

t-test Result	OneMax				Plateau				RoyalRoad				Deceptive			
$\tau = 50, \rho \Rightarrow$	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9
AHMA0 − AHMA1	−	−	−	−	−	−	−	−	∼	∼	∼	−	−	−	−	−
AHMA0 − AHMA2	+	+	+	+	+	+	+	+	∼	∼	∼	∼	+	+	+	+
AHMA1 − AHMA2	+	+	+	+	+	+	+	+	∼	+	∼	+	+	+	+	+

while the number of generated immigrants (num_{im}) would decrease as a result of $num_{im} = num\_aepg - num_{ls}$. The experimental results show that this scheme dominated by LS is a good choice to keep balance between LS and diversity maintaining.

Second, IMAHMA2 is always beaten by IMAHMA0 and IMAHMA1 on most dynamic problems. In IMAHMA2, the number of generated immigrants is firstly determined based on the convergence status of population and then the value of num_{ls} is calculated from the formula $num_{ls} = num\_aepg - num_{im}$. When the index ($\xi$) is lower than a threshold, the immigrants' number (num_{im}) would increase in order to improve the diversity level of population. Obviously, the idea behind this scheme is to maintain a spread-out population. However, the experimental results indicate that this simple scheme cannot achieve the original purpose.

Finally, IMAHMA0 exhibits the good performance on the test problems, which validate the necessary of hybridizing LS and diversity maintaining scheme in MAs for DOPs.

6.4.4 Experimental Study on Comparing the Proposed Algorithm with Several Peer GAs on DOPs

In the final experiments, we compare the performance of IMAHMA with several peer GAs on the DOPs constructed in Section 6.3. These peer GAs are SGAr, RIGA, EIGA and DIGA, as described in Section 6.4.1. The experimental results are presented in Table 6.6 to Table 6.9, respectively. The corresponding statistical results are given in Table 6.10. From these tables, several results can be observed and analyzed as follows.

First, IMAHMA always outperforms other peer algorithm on most dynamic problems and underperforms some of these GAs on some dynamic problems when the environment changes slowly, i.e., when $\tau = 50$ or $\tau = 100$. When the environment changes quickly, i.e., when $\tau = 10$, IMAHMA can always locate the optimum (maybe local optimum) more quickly than other GAs because LS has a strong exploitation capacity. This is why IMAHMA performs the best on all dynamic problems with $\tau = 10$. When $\tau = 50$ or $\tau = 100$, IMAHMA performs worse than SGAr on dynamic Plateau and RoyalRoad problems with $\rho = 0.5$. This happens because the random environment always requires algorithms to maintain a sufficient population diversity (see the relevant analysis in Section 6.4.3) and the restart scheme

Table 6.6 Experimental results with respect to overall offline performance of IMAHMA and peer GAs on dynamic OneMax problems

Dynamics		Algorithms				
τ	ρ	IMAHMA	SGAr	EIGA	RIGA	DIGA
10	0.1	91.78±0.60	70.76±0.45	89.50±0.81	87.97±0.66	89.35±0.91
10	0.2	86.38±0.84	70.68±0.36	81.42±0.85	80.58±1.03	81.14±1.09
10	0.5	78.93±0.93	70.66±0.28	71.66±1.73	71.83±0.77	71.52±1.41
10	0.9	91.70±0.42	70.68±0.34	88.32±2.21	87.67±0.84	89.31±0.82
50	0.1	97.11±0.26	91.02±0.14	97.09±0.22	95.06±0.34	97.23±0.25
50	0.2	94.85±0.35	91.10±0.16	94.44±0.45	90.89±0.54	94.50±0.42
50	0.5	92.92±0.34	91.05±0.13	90.05±0.72	89.41±0.32	90.22±0.63
50	0.9	97.11±0.16	91.00±0.31	97.14±0.46	97.08±0.30	97.26±0.24
100	0.1	98.55±0.08	95.50±0.08	98.58±0.06	97.48±0.14	98.59±0.09
100	0.2	97.43±0.13	95.52±0.07	97.21±0.18	95.37±0.24	97.17±0.19
100	0.5	96.46±0.15	95.51±0.08	95.13±0.34	94.69±0.12	95.15±0.33
100	0.9	98.54±0.11	95.52±0.05	98.54±0.25	97.47±0.18	98.60±0.11

Table 6.7 Experimental results with respect to overall offline performance of IMAHMA and peer GAs on dynamic Plateau problems

Dynamics		Algorithms				
τ	ρ	IMAHMA	SGAr	EIGA	RIGA	DIGA
10	0.1	83.03±0.95	47.96±0.60	76.17±2.13	77.53±1.05	76.86±2.72
10	0.2	72.91±1.64	47.77±0.75	63.75±2.02	64.57±1.36	64.31±2.08
10	0.5	57.52±2.02	48.03±0.62	48.12±2.17	49.25±0.87	47.68±1.32
10	0.9	83.46±1.12	47.89±0.72	76.59±4.83	77.79±1.21	78.12±2.74
50	0.1	93.40±0.37	82.99±0.34	92.91±0.83	90.73±0.51	93.01±0.70
50	0.2	87.56±0.96	83.00±0.30	86.45±0.80	82.92±0.83	86.67±1.22
50	0.5	81.42±0.89	82.99±0.40	77.42±1.41	80.37±0.67	78.19±1.18
50	0.9	93.33±0.53	82.81±0.38	92.71±1.94	90.74±0.71	92.93±0.79
100	0.1	96.71±0.32	91.46±0.16	96.24±0.60	95.37±0.33	96.05±0.88
100	0.2	93.32±0.63	91.52±0.17	92.55±0.87	91.13±0.53	92.43±0.93
100	0.5	90.72±0.59	91.51±0.13	86.51±1.19	90.22±0.38	87.18±1.41
100	0.9	96.72±0.31	91.52±0.15	96.07±0.97	95.37±0.36	96.17±0.80

in SGAr can introduce the maximum diversity into the population. The reason why SGAr outperforms IMAHMA only on the Plateau and RoyalRoad problems lies in the intrinsic characteristics of these problems. The OneMax problem is simply unimodal, which is very suitable for a HC search in IMAHMA. Both the Plateau and RoyalRoad problems have higher-order building blocks, which take a HC search much more time to achieve. The Deceptive problem may mislead SGAr's evolution due to the existence of deceptive attractor, which can be escaped from by IMAHMA. IMAHMA is also beaten by DIGA on the dynamic OneMax problems and Deceptive problems with $\rho = 0.5$ and $\tau = 50$ or 100. This is because DIGA can especially fit such an acutely-changing environment and the dual-based immigrants can maintain

Table 6.8 Experimental results with respect to overall offline performance of IMAHMA and peer GAs on dynamic RoyalRoad problems

Dynamics		Algorithms				
τ	ρ	IMAHMA	SGAr	EIGA	RIGA	DIGA
10	0.1	67.27±2.39	32.50±0.87	53.89±4.71	62.30±2.30	53.03±5.37
10	0.2	52.48±2.33	32.52±0.73	43.44±4.53	47.95±1.39	41.75±3.53
10	0.5	38.83±1.53	32.31±0.76	31.67±1.91	33.98±1.48	32.41±1.85
10	0.9	66.09±2.72	32.69±0.73	52.25±4.97	62.29±2.83	54.87±4.79
50	0.1	85.70±1.38	69.32±0.73	76.39±4.13	83.94±1.24	75.84±4.28
50	0.2	74.48±1.10	69.40±1.03	66.33±2.79	72.27±1.54	66.62±3.41
50	0.5	63.40±1.99	69.38±0.83	53.17±2.20	65.58±1.79	53.05±2.44
50	0.9	85.26±1.31	69.10±1.02	77.38±4.31	83.30±1.41	76.28±3.27
100	0.1	91.31±0.98	84.42±0.48	82.56±2.98	90.15±0.99	82.39±4.05
100	0.2	82.64±1.44	84.33±0.43	74.04±2.15	82.13±1.34	73.36±1.96
100	0.5	76.07±2.00	84.39±0.58	62.13±2.17	79.80±1.67	62.55±2.58
100	0.9	91.38±0.79	84.30±0.42	82.99±5.60	90.50±1.10	82.79±3.47

Table 6.9 Experimental results with respect to overall offline performance of IMAHMA and peer GAs on dynamic Deceptive problems

Dynamics		Algorithms				
τ	ρ	IMAHMA	SGAr	EIGA	RIGA	DIGA
10	0.1	72.78±2.04	53.13±0.56	69.15±1.75	64.36±1.55	71.77±1.75
10	0.2	66.05±0.98	52.92±0.50	62.94±1.44	57.88±1.26	63.24±1.38
10	0.5	61.73±0.76	53.18±0.47	56.70±1.01	53.28±0.78	57.55±1.05
10	0.9	73.04±1.54	53.23±0.64	69.18±1.19	64.02±1.19	71.63±1.36
50	0.1	83.91±0.99	66.06±0.57	78.07±1.23	75.47±1.66	84.58±0.49
50	0.2	81.50±0.73	66.02±0.62	75.04±1.10	70.49±1.03	81.79±0.49
50	0.5	80.14±0.60	66.17±0.52	73.77±0.71	68.06±0.82	79.61±0.49
50	0.9	83.64±0.51	66.23±0.70	78.20±1.66	75.28±1.90	84.58±0.46
100	0.1	87.14±0.86	73.85±0.54	79.07±1.09	77.86±1.61	87.21±0.45
100	0.2	86.07±0.55	73.96±0.58	77.45±1.10	75.83±1.14	86.20±0.54
100	0.5	85.95±0.34	73.94±0.63	77.41±0.73	74.89±0.76	85.41±0.40
100	0.9	86.93±0.55	73.90±0.43	78.61±1.91	78.51±1.81	87.04±0.57

a very high fitness level on the OneMax and Deceptive problems. The good performance of IMAHMA over other peer GAs shows that our investigated IMAHMA has a strong robustness and adaptivity in dynamic environments.

Second, on dynamic OneMax and Plateau problems EIGA always outperforms SGAr and RIGA when ρ is set to 0.1 or 0.2, but underperforms them when the value of ρ is set to 0.5 or 0.9. On dynamic RoyalRoad and Deceptive problems, the situations become a little different. EIGA performs better than RIGA on dynamic RoyalRoad problems just when $\tau = 10$ and better than both SGAr and RIGA on all dynamic Deceptive problems. This happens because the elitism-based immigrants scheme can introduce higher fitness individuals, which can adapt better to the current

Table 6.10 The t-test results of comparing the overall offline performance of AHMA and peer EAs on dynamic test problems

t-test Result	OneMax				Plateau				RoyalRoad				Deceptive			
$\tau = 10, \rho \Rightarrow$	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9
IMAHMA − SGAr	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IMAHMA − EIGA	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IMAHMA − RIGA	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IMAHMA − DIGA	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
SGAr − EIGA	−	−	−	−	−	−	∼	−	−	−	∼	−	−	−	−	−
SGAr − RIGA	−	−	−	−	−	−	−	−	−	−	−	−	−	−	∼	−
SGAr − DIGA	−	−	−	−	−	−	∼	−	−	−	∼	−	−	−	−	−
EIGA − RIGA	+	+	∼	∼	−	∼	−	∼	−	−	−	−	+	+	+	+
EIGA − DIGA	∼	∼	∼	−	∼	∼	∼	∼	∼	∼	∼	∼	−	∼	−	−
RIGA − DIGA	−	∼	∼	−	∼	∼	+	∼	+	+	+	+	−	−	−	−
$\tau = 50, \rho \Rightarrow$	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9
IMAHMA − SGAr	+	+	+	+	+	+	−	+	+	+	−	+	+	+	+	+
IMAHMA − EIGA	∼	+	+	∼	+	+	+	∼	+	+	+	+	+	+	+	+
IMAHMA − RIGA	+	+	+	+	+	+	+	+	+	+	−	+	+	+	+	+
IMAHMA − DIGA	∼	+	+	−	+	+	+	+	+	+	+	+	−	∼	+	−
SGAr − EIGA	−	−	+	−	−	−	+	−	−	+	+	−	−	−	−	−
SGAr − RIGA	−	+	+	−	−	∼	+	−	−	−	+	−	−	−	−	−
SGAr − DIGA	−	−	+	−	−	−	+	−	−	+	+	−	−	−	−	−
EIGA − RIGA	+	+	+	+	+	+	−	+	−	−	−	−	+	+	+	+
EIGA − DIGA	∼	∼	∼	∼	∼	∼	−	∼	∼	∼	∼	∼	−	−	−	−
RIGA − DIGA	−	−	−	−	−	−	−	−	+	+	+	+	−	−	−	−
$\tau = 100, \rho \Rightarrow$	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9	0.1	0.2	0.5	0.9
IMAHMA − SGAr	+	+	+	+	+	+	−	+	+	−	−	+	+	+	+	+
IMAHMA − EIGA	∼	+	+	∼	+	+	+	+	+	+	+	+	+	+	+	+
IMAHMA − RIGA	+	+	+	+	+	+	+	+	+	∼	−	+	+	+	+	+
IMAHMA − DIGA	∼	+	+	∼	+	+	+	+	+	+	+	+	∼	∼	+	∼
SGAr − EIGA	−	−	+	−	−	−	+	−	+	+	+	∼	−	−	−	−
SGAr − RIGA	−	+	+	−	−	+	+	−	−	+	+	−	−	−	−	−
SGAr − DIGA	−	−	+	−	−	−	+	−	+	+	+	+	−	−	−	−
EIGA − RIGA	+	+	+	+	+	+	−	+	−	−	−	−	+	+	+	∼
EIGA − DIGA	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	−	−	−	−
RIGA − DIGA	−	−	−	−	−	−	+	−	+	+	+	+	−	−	−	−

environment, into EIGA's population on dynamic OneMax and Plateau problems when the environment changes slightly, on dynamic RoyalRoad problems when the environment changes quickly, and on all dynamic Deceptive problems due to the intrinsic characteristics of these four kinds of functions.

Third, RIGA always performs better than SGAr when the value of ρ is small on most dynamic test problems. This is because a new environment is close to the previous one when the value of ρ is very small. Therefore, introducing a portion of random individuals into the population as done in RIGA may be more beneficial than re-initializing the whole population as done in SGAr.

Fourth, DIGA exhibits good performance on OneMax, Plateau, and Deceptive problems although DIGA is beaten by SGAr and RIGA on these dynamic problems when $\rho = 0.5$. This also confirms the exception of the dual-based immigrants scheme for GAs in dynamic environments. DIGA performs better than the other GAs on all dynamic Deceptive problems. The reason lies in that the dualism mechanism may help DIGA react to significant environmental changes rapidly and also enable it to escape from the deceptive attractor in the Deceptive function.

Finally, the environmental parameters affect the performance of algorithms. The performance of all algorithms increases when the value of τ increase from 10 to 50 to 100. It is easy to understand. When τ becomes larger, algorithms have more time to find better solutions before the next change. The effect of the changing severity parameter ρ is different. For example, when τ is fixed, the performance curve of IMAHMA always declines when ρ increases from 0.1 to 0.2 to 0.5, but rises when ρ increases from 0.5 to 0.9.

In order to better understand the experimental results, we make a deeper look into the dynamic behavior of these algorithms. The dynamic behavior of different algorithms with respect to the running offline performance is shown in Fig. 6.12 to Fig. 6.15, where τ is set to 50 and ρ is set to 0.1, 0.2, 0.5, and 0.9, respectively. From these figures, it can be easily observed that for the dynamic periods SGAr always performs almost the same as it did for the stationary period (the first 50 generations) and IMAHMA always outperforms other peer GAs for the stationary period on all test problems while their dynamic behaviors are different on different dynamic problems.

On the OneMax and Plateau problems (see Figs. 6.12 and 6.13), the dynamic behavior of IMAHMA for each dynamic period outperforms that for the stationary period when ρ is very small. When $\rho = 0.5$, IMAHMA exhibits almost the same performance in both the stationary and dynamic periods. When the value of ρ increases to 0.9, the performance of IMAHMA in the dynamic period upgrades consistently. This is because that the proposed AHC operator can help IMAHMA achieve the optimum quickly once the population can "find" in the area where the optimum is located. The new optimum may be close to the previous one when the environment changes slightly, while the immigrants scheme in IMAHMA can help the population re-locate to the changing optimum quickly when a significant environmental change occurs. The dynamic behavior of both RIGA and EIGA is affected by the value of ρ. With the increment of dynamic periods, their performance upgrades consistently when $\rho = 0.1$, while their behavior for the dynamic periods underperforms that for the stationary period when $\rho = 0.5$ or 0.9. The behavior of DIGA is very similar to the behavior of RIGA and EIGA when ρ is not very large. However, DIGA exhibits very nice adaptability to a significantly changing environment. Its behavior for the dynamic periods outperforms that for the stationary period when ρ=0.9. This

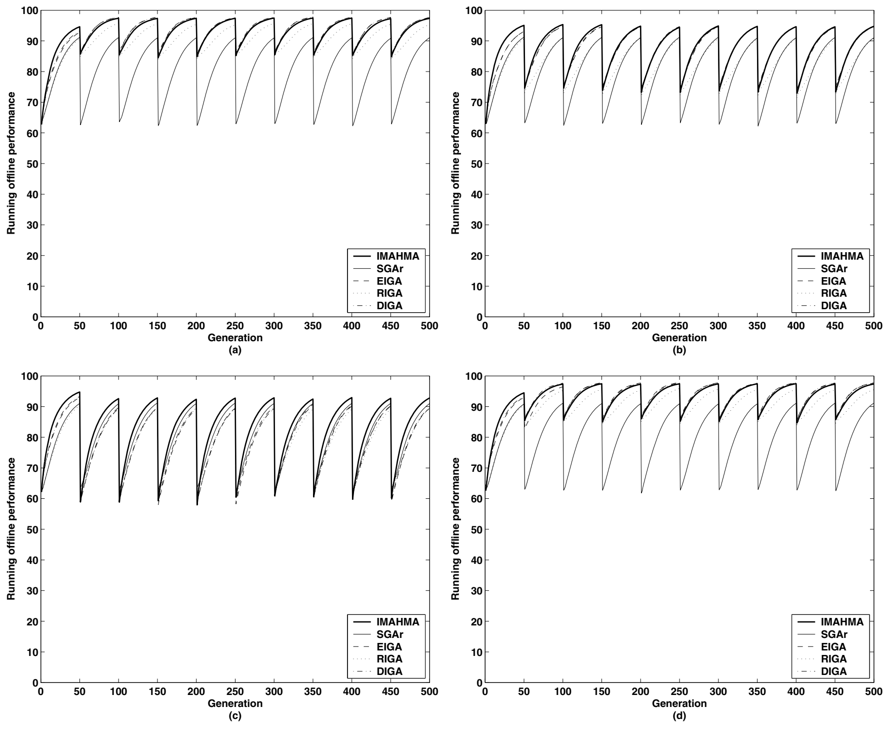

Fig. 6.12 Dynamic behavior of IMAHMA and peer GAs on dynamic OneMax problems with $\tau = 50$ and ρ is set to: (a) $\rho = 0.1$, (b) $\rho = 0.2$, (c) $\rho = 0.5$, and (d) $\rho = 0.9$

happens because the dual-based immigrants scheme can help DIGA re-locate to the changing optimum quickly when a sharp environmental change occurs.

On the RoyalRoad and Deceptive problems (see Fig. 6.14), with the increment of dynamic periods, IMAHMA's performance drops a little when $\rho = 0.5$, while rises when $\rho = 0.1$, 0.2 and 0.9. The reason lies in that IMAHMA does not find the optimum in the stationary period on these two problems. When the environment changes slightly or very significantly, IMAHMA always reruns from the starting points with a higher fitness in the dynamic periods than that in the stationary period, while when $\rho = 0.5$, IMAHMA can only obtain worse starting points in the dynamic periods.

6.5 Conclusions and Future Work

In this chapter, the application of MAs with an adaptive hill climbing strategy for dynamic optimization problems is investigated. In the proposed MA, two local search methods, a greedy crossover-based hill climbing and a steepest mutation-based hill climbing, are used to refine the individual that is selected for local improvements.

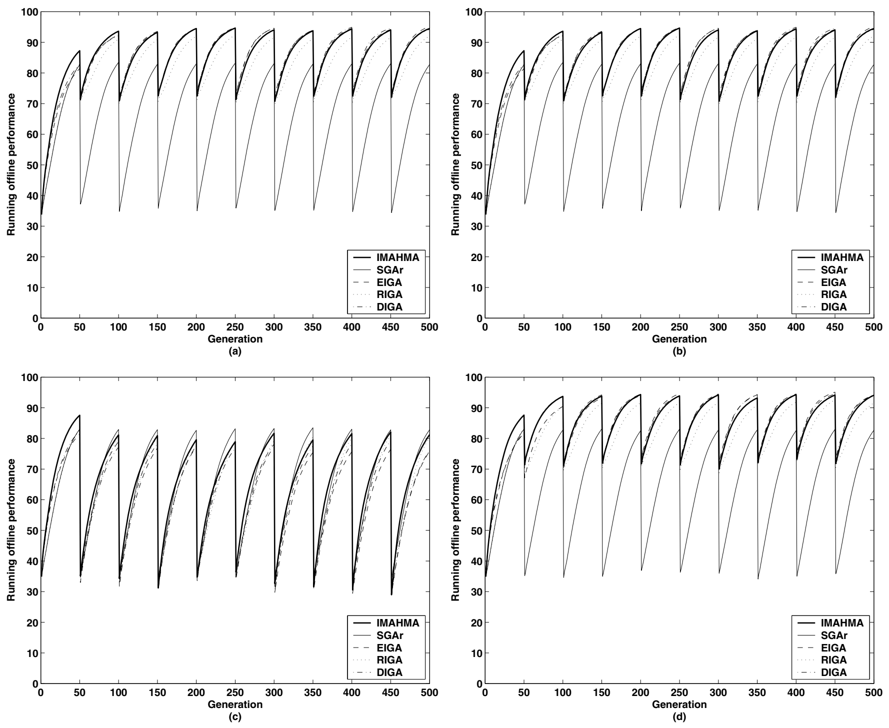

Fig. 6.13 Dynamic behavior of IMAHMA and peer GAs on dynamic Plateau problems with $\tau = 50$ and ρ is set to: (a) $\rho = 0.1$, (b) $\rho = 0.2$, (c) $\rho = 0.5$, and (d) $\rho = 0.9$

A learning mechanism, which gives the more effective LS operator greater chance for the later individual refinement, is introduced in order to execute a robust local search. To maintain a sufficient population diversity for the algorithms to adapt well to the environmental changes, a new immigrants scheme, where the immigrants can be generated from mutating an elite individual with a probability adaptively, is introduced into our proposed MA. In order to keep the balance between LS and diversity maintaining with respect to the extra computation cost, two different dominated schemes are also discussed in this paper.

In order to test the performance of the proposed MA for DOPs, a series of experimental studies have been carried out based on a set of constructed dynamic test problems. From the experimental results, we can draw the following conclusions on the dynamic test problems.

First, MAs enhanced by suitable diversity methods can exhibit a better performance in dynamic environments. For most dynamic test problems, IMAHMA always outperforms other peer GAs.

Second, the immigrants scheme is efficient for improving the performance of MAs in dynamic environments. However, different immigrants schemes have different effects in different dynamic environments. The elitism-based immigrants

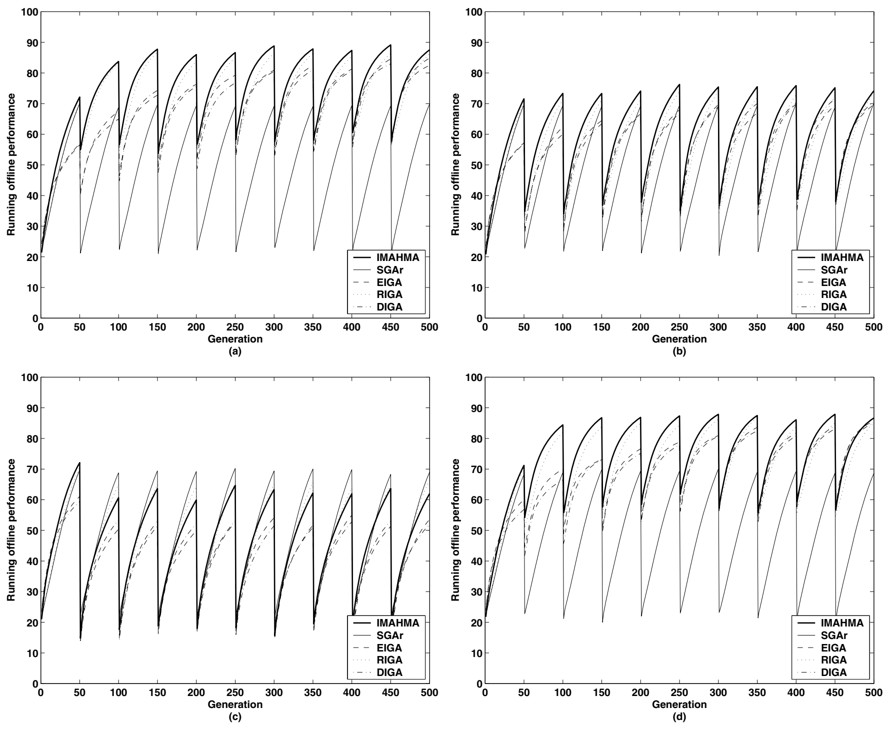

Fig. 6.14 Dynamic behavior of IMAHMA and peer GAs on dynamic RoyalRoad problems with $\tau = 50$ and ρ is set to: (a) $\rho = 0.1$, (b) $\rho = 0.2$, (c) $\rho = 0.5$, and (d) $\rho = 0.9$

scheme is suitable for the slightly-changing environments, the random immigrants scheme performs better when the environmental severity $\rho = 0.5$, and the dualism-based immigrants does better when the environment involves significant changes (i.e., $\rho = 0.9$). The proposed immigrants scheme in Section 6.2.3 is a good choice that generalizes three different immigrants schemes within a common framework.

Third, the effect of LS operators is problem dependent. The AHC strategy can help MAs execute a robust individual refinement since it employs multiple LS operators under the mechanism of cooperation and competition.

Fourth, the difficulty of DOPs depends on the environmental dynamics, including the severity and speed of changes and the difficulty of the base stationary problems. According to our experiments, MAs perform better with the increasing of the frequency of changes, and the effect of the severity of changes is problem dependent.

Generally speaking, the experimental results indicate that the proposed MA, where the adaptive hill climbing operator is used as a local search technique for individual refinement, with adaptive dual mapping and triggered random immigrants schemes, seems a good EA optimizer for DOPs.

For the future work, it is straightforward to introduce other mechanisms, such as memory-based methods [43] and multi-population approaches [30], into MAs

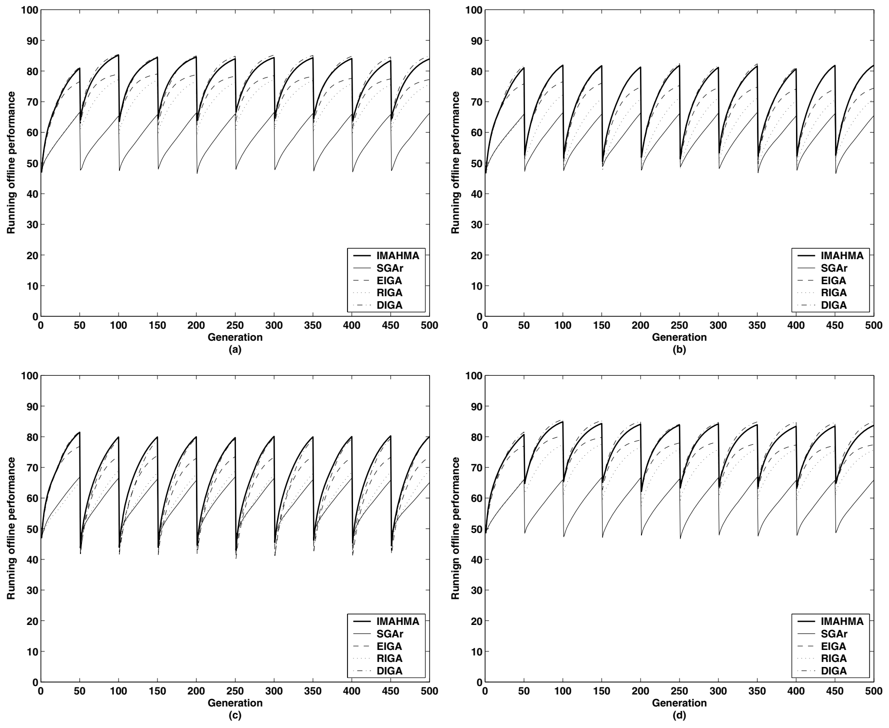

Fig. 6.15 Dynamic behavior of IMAHMA and peer GAs on dynamic Deceptive problems with $\tau = 50$ and ρ is set to: (a) $\rho = 0.1$, (b) $\rho = 0.2$, (c) $\rho = 0.5$, and (d) $\rho = 0.9$

for DOPs. Another interesting research work is to extend the proposed immigrants scheme to other EAs and examine their performance in dynamic environments. In addition, it is also valuable to carry out the sensitivity analysis on the effect of parameters, e.g., α, β, Δ, θ_0, λ_0, and λ_1, on the performance of proposed MAs in the future.

Acknowledgements. The work by Hongfeng Wang was supported by the National Nature Science Foundation of China (NSFC) under Grant 71001018, the National Innovation Research Community Science Foundation of China under Grant 60521003, the China Postdoctoral Science Foundation Under Grant 2012T50266, and the Fundamental Research Funds for the Central Universities Grant N110404019 and Grant N110204005. The work by Shengxiang Yang was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1 and partially by the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, China.

References

[1] Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-94-163, Carnegie Mellon University, USA (1994)

[2] Bosman, P.A.N., Poutré, H.L.: Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In: Proc. 2007 Genetic and Evol. Comput. Conf., pp. 1165–1172 (2007)

[3] Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proc. 1999 IEEE Congr. Evol. Comput., pp. 1875–1882 (1999)

[4] Branke, J., Kaußler, T., Schmidth, C., Schmeck, H.: A multi-population approach to dynamic optimization problems. In: Proc. 4th Int. Conf. Adaptive Comput. Des. Manuf., pp. 299–308 (2000)

[5] Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environment. Tech. Rep. AIC-90-001, Naval Research Laboratory, Washington, USA (1990)

[6] Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3(2), 124–141 (1999)

[7] Eriksson, R., Olsson, B.: On the behavior of evolutionary global-local hybrids with dynamic fitness functions. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 13–22. Springer, Heidelberg (2002)

[8] Eriksson, R., Olsson, B.: On the Performance of Evolutionary Algorithms with life-time adaptation in dynamic fitness landscapes. In: Proc. 2004 IEEE Congr. Evol. Comput., pp. 1293–1300 (2004)

[9] Gallardo, J.E., Cotta, C., Ferndez, A.J.: On the hybridization of memetic algorithms with branch-and-bound techniques. IEEE Trans. Syst., Man, and Cybern.-Part B: Cybern. 37(1), 77–83 (2007)

[10] Goh, C.K., Tan, K.C.: A competitive-cooperation coevolutionary paradigm for dynamic multi-objective optimization. IEEE Trans. Evol. Comput. 13(1), 103–127 (2009)

[11] Goldberg, D.E., Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Proc. 2nd Int. Conf. on Genetic Algorithms, pp. 59–68 (1987)

[12] Grefenstette, J.J.: Genetic algorithms for changing environments. In: Proc. 2nd Int. Conf. Parallel Problem Solving From Nature, pp. 137–144 (1992)

[13] Hatzakis, I., Wallace, D.: Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In: Proc. 2006 Genetic and Evol. Comput. Conf., pp. 1201–1208 (2006)

[14] Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans. Evol. Comput. 7(2), 204–223 (2003)

[15] Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments–A survey. IEEE Trans. Evol. Comput. 9(3), 303–317 (2005)

[16] Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans. Evol. Comput. 9(5), 474–487 (2005)

[17] Lau, T.L., Tsang, E.P.K.: Applying a mutation-based genetic algorithm to processor configuration problems. In: Proc. 8th IEEE Conf. on Tools with Artif. Intell., pp. 17–24 (1996)

[18] Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing. Evol. Comput. 12(3), 273–302 (2004)
[19] Liu, D., Tan, K.C., Goh, C.K., Ho, W.K.: A multiobjective memetic algorithm based on particle swarm optimization. IEEE Trans. Syst., Man, and Cybern.-Part B: Cybern. 37(1), 42–50 (2007)
[20] Liu, B., Wang, L., Jin, Y.H.: An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Trans. Syst., Man, and Cybern.-Part B: Cybern. 37(1), 18–27 (2007)
[21] Man, S., Liang, Y., Leung, K.S., Lee, K.H., Mok, T.S.K.: A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. IEEE Trans. Syst., Man, and Cybern.-Part B: Cybern. 37(1), 84–91 (2007)
[22] Neri, F., Toivanen, J., Cascella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ACM Trans. Comput. Biology and Bioinform. 4(2), 264–278 (2007)
[23] Neri, F., Toivanen, J., Makinen, A.R.E.: An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV. Applied Intell. 27(3), 219–235 (2007)
[24] Nguyen, T.T., Yao, X.: Dynamic time-linkage problems revisited. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 735–744. Springer, Heidelberg (2009)
[25] Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evol. Comput. 6, 1–24 (2012)
[26] Ong, Y.S., Keane, A.J.: Meta-lamarckian learning in memetic algorithms. IEEE Trans. Evol. Comput. 8(2), 99–110 (2004)
[27] Oppacher, F., Wineberg, M.: The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In: Proc. 1999 Genetic and Evol. Comput. Conf., vol. 1, pp. 504–510 (1999)
[28] O'Reilly, U.M., Oppacher, F.: Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 397–406. Springer, Heidelberg (1994)
[29] O'Reilly, U.M., Oppacher, F.: Hybridized crossover-based search techniques for program discovery. In: Proc. 1995 IEEE Int. Conf. Evol. Comput., pp. 573–578 (1995)
[30] Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Comput. 10(4), 440–458 (2006)
[31] Simões, A., Costa, E.: Evolutionary algorithms for dynamic environments: Prediction using linear regression and markov chains. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 306–315. Springer, Heidelberg (2008)
[32] Simões, A., Costa, E.: Improving prediction in evolutionary algorithms for dynamic environments. In: Proc. 2009 Genetic and Evol. Comput. Conf., pp. 875–882 (2009)
[33] Smith, J.E.: Coevolving memetic algorithms: A review and progress report. IEEE Trans. Syst., Man and Cybern.-Part B: Cybern. 37(1), 6–17 (2007)
[34] Talbi, E.G., Bachelet, V.: Cosearch: A parallel cooperative metaheuristic. J. of Mathematical Modelling and Algorithms 5(1), 5–22 (2006)
[35] Tang, J., Lim, M.H., Ong, Y.S.: Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. Soft Comput. 11(10), 957–971 (2007)
[36] Tang, M., Yao, X.: A memetic algorithm for VLSI floor planning. IEEE Trans. Syst., Man and Cybern.-Part B: Cybern. 37(1), 62–69 (2007)

[37] Uyar, A.S., Harmanci, A.E.: A new population based adaptive dominance change mechanism for diploid genetic algorithms in dynamic environments. Soft Comput. 9(11), 803–815 (2005)
[38] Vavak, F., Fogarty, T.C., Jukes, K.: Adaptive combustion balancing in multiple burner boilers using a genetic algorithm with variable range of local search. In: Proc. 7th Int. Conf. on Genetic Algorithms, pp. 719–726 (1996)
[39] Wang, H., Wang, D.: An improved primal-dual genetic algorithm for optimization in dynamic environments. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006, Part III. LNCS, vol. 4234, pp. 836–844. Springer, Heidelberg (2006)
[40] Wang, H., Wang, D., Yang, S.: Triggered memory-based swarm optimization in dynamic environments. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 637–646. Springer, Heidelberg (2007)
[41] William, E.H., Krasnogor, N., Smith, J.E. (eds.): Recent Advances in Memetic Algorithms. Springer, Heidelberg (2005)
[42] Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm. In: Proc. 2003 IEEE Congr. Evol. Comput., vol. 3, pp. 2246–2253 (2003)
[43] Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., et al. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 788–799. Springer, Heidelberg (2006)
[44] Yang, S.: Genetic algorithms with elitism-based immigrants for changing optimization problems. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 627–636. Springer, Heidelberg (2007)
[45] Yang, S., Jiang, Y., Nguyen, T.T.: Metaheuristics for dynamic combinatorial optimization problems. IMA J. of Management Mathematics (2012), doi:10.1093/imaman/DPS021
[46] Yang, S., Ong, Y.S., Jin, Y. (eds.): Evolutionary Computation in Dynamic and Uncertain Environments. Springer, Heidelberg (2007)
[47] Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)
[48] Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. Evol. Comput. 12(5), 542–561 (2008)
[49] Zhou, Z., Ong, Y.S., Lim, M.H.: Memetic algorithm using multi-surrogates for computationally expensive optimization problems. Soft Comput. 11(9), 873–888 (2007)
[50] Zhu, Z., Ong, Y.S., Dash, M.: Wrapper-filter feature selection algorithm using a memetic framework. IEEE Trans. Syst., Man and Cybern.-Part B: Cybern. 37(1), 70–76 (2007)