

# Agent based Evolutionary Dynamic Optimization

Yang Yan <sup>1</sup> , Shengxiang Yang <sup>2</sup>, Dazhi Wang<sup>1</sup> and Dingwei Wang<sup>1</sup>

1. School of Information Science and Engineering

Northeastern University, Shenyang 110004, China.

yanyangmail@163.com; dwwang@mail.neu.edu.cn; wongdz@gmail.com

2. Department of Computer Science, University of Leicester

University Road, Leicester LE1 7RH, United Kingdom.

s.yang@mcs.le.ac.uk

**Summary** Agent-based Evolutionary Search (AES) has attracted a growing interest from the evolutionary computation community in recent years due to its robust ability in solving large scale problems, ranging from online trading, disaster response, modeling social structures to financial investment planning. In order to solve these problems, a great variety of intelligent techniques have been developed to improve the framework and efficiency of AES. This chapter investigates an agent based evolutionary search algorithm in which the agents are updated and co-evolve to track dynamic optimum by imitating the exhibited feature of living organism. In the proposed algorithm, all agents live in a lattice like environment, where each agent is fixed on a lattice point. In order to increase the predefined energy function, individual agent can compete with its neighbors and also can acquire knowledge through cumulative information. For the purpose of maintaining the diversity of the population, random immigrants and adaptive primal dual mapping schemes are incorporated. Simulation experiments on a set of dynamic benchmark problems show the proposed AES algorithm can yield a better performance on dynamic optimization problems (DOPs) in comparison with several peer algorithms.

## 1 Introduction

Agent based system which is composed of multiple interacting agents has generated lots of excitement in recent years because it has been suggested as a promising technique for conceptualizing and solving various optimization problems [12, 17, 22, 21, 28]. This promising technique is particularly attractive for solving operational problems in business and logistics environments, such as the production planning problems, transportation and distribution problems, and resource allocation problems [10, 10, 18, 23]. Zhong et al. [40] integrated agent systems with Genetic Algorithms (GAs) to form a new algorithm for solving the global numerical optimization problem.

In recent years, AES has been applied widely to solve stationary optimization problems where the fitness curve or the objective function maintain unchanged during the searching process of AES. However most real-world optimization problems are dynamic since the objective function, environmental parameter and/or constraint conditions maybe change over time. For example, in job scheduling production systems, re-scheduling of the jobs is often required due to change of the specification of jobs or malfunction of the machines. For these DOPs, the goal of an algorithm is no longer to find an optimal solution but to track the moving optima in the search space. In order to enhance the performance of Evolutionary Algorithms (EAs) on DOPs, many researchers have developed a number of approaches, including diversity-increasing methods [8], diversity-keeping methods [2, 15], memory-based approaches [4], and multi-population approaches [3, 5].

In this chapter, a hybrid Agent-based Evolutionary Search algorithm is proposed. In the proposed AES, a population of agents that represent potential solutions are located in a lattice like environment and accomplish their update via competing in a local neighborhood or learning strategy based on the statistical feedback information of the whole system. Similar to other EAs, AES may gradually converge in the search space during the run, especially when the environment has been stationary

for some time. In order to address such problem, two diversity maintaining, random immigrants and adaptive dual mapping, are introduced into our AES to improve its performance for DOPs.

The rest of this chapter is organized as follows. Section 2 describes the AES algorithm in details. In Section 3, the dynamic testing suite is presented. Section 4 reports the experimental results. Finally, Section V concludes this chapter.

## 2 Proposed Agent based Evolutionary Search Algorithm

### 2.1 The Framework of AES

Multi-agent system, also called ‘self organized system’ is a computational system in which multiple interacting intelligent agents work together to solve difficult problems which may be impossible for an individual agent. In a nutshell, the agent in a multi-agent system is a physical or virtual entity which has several important characteristics. According to [21], four elements should be defined before an agent-based system is used to solve problem:

- First, it is able to live and act in the environment.
- Second, it is able to sense its local environment.
- Third, it is driven by certain purposes.
- Fourth, it has some reactive behaviors.

As can be seen, the meaning of an agent is quite comprehensive, and what an agent represents is different from each other for specific optimization problems. In this chapter, an AES based algorithm is proposed to investigate the dynamic 0-1 optimization problems. Obviously, each agent is a 0-1 array, which represents a candidate solution. The energy value of an agent may be equal to the value of the objective function. More detailed definitions can be described as follows:

*Definition 1:* An agent  $L$  represents a candidate solution to the optimization problem in hand, and can be expressed as a 0-1 array:

$$L = (L^1, L^2, \dots, L^n), L^i = 0 \text{ or } 1, 1 \leq i \leq n, \quad (1)$$

where  $n$  represents the scale of the problem. The value of its energy  $E(L)$  is equal to the value of the objective function  $f(L)$ , that is,  $E(L) = f(L)$ .

In order to realize the local perceptivity of agents conveniently, the environment is organized as a ring-shaped lattice-like structure. All agents live in a lattice-like environment, which is called an agent lattice  $LL$ . The size of environment is  $L_{size} \times L_{size}$ .

Each agent is fixed on a lattice-point and only interacts with its neighbors. Suppose that the agent located at  $(i, j)$  is represented as  $L_{i,j}$ . And the agents which interact with  $L_{i,j}$  could be decided by the apperceive range  $R_s$ . Thus, the neighbor agents of  $L_{i,j}$  could be calculated as follows:

$$L_{k,l}, i - R_s \leq k \leq i + R_s, j - R_s \leq l \leq j + R_s. \quad (2)$$

Due to the ring-shaped lattice structure, if  $k < 1$ , then  $k = k + L_{size}$ ; if  $k > L_{size}$ , then  $k = k - L_{size}$ ; if  $l < 1$ , then  $l = l + L_{size}$ ; if  $l > L_{size}$ , then  $l = l - L_{size}$ . The agents that interact with  $L_{i,j}$  are called its neighbor  $N_{i,j}$ .

Therefore, the agent lattice can be represented as the one in Fig. 1. Each lattice represents an agent, the data in a circle represents its position in the lattice, and two agents can interact with each other if and only if there is a line connecting them. In traditional genetic algorithms (GAs), those individuals that will generate offspring are usually selected from all individuals according to their fitness. Therefore, the global fitness distribution of a population must be determined. But in nature, a global selection does not exist, and the global fitness distribution cannot be determined either.

In fact, the natural selection in real world only occurs in a local environment, and each individual can only interact with those around it. That is, in some phases, the natural evolution is just a kind

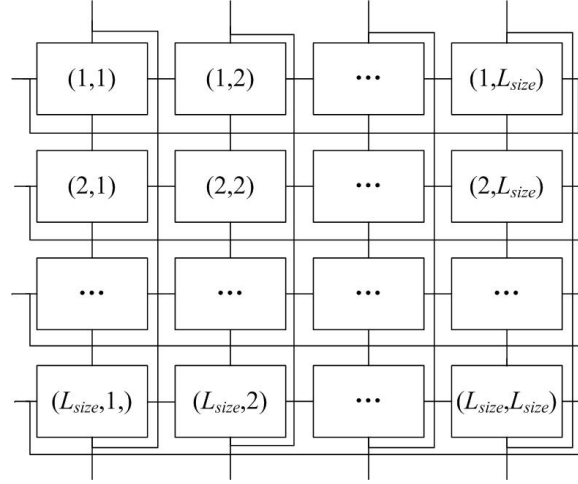


Figure 1: Model of the agent lattice

of local phenomenon. The information can be shared globally only after a process of diffusion. In the agent lattice, in order to achieve their purposes, agents will compete or cooperate with each other so that they can gain more resources. Since each agent can only sense its local environment, its behaviors of competition and cooperation can only take place among the agent and its neighbors. Simultaneously, the feedback of the best agent can usually influence the behavior of the agents. An agent interacts with its neighbors so that information is transferred to them, and the information feedback from the Lattice can help agents to learn. In such a manner, the information is diffused to the whole agent lattice. As can be seen, the model of the agent lattice is more close to the real evolutionary mechanism in nature than the model of the population in traditional GAs. In summary, the general AES algorithm can be illustrated in Fig. 2.

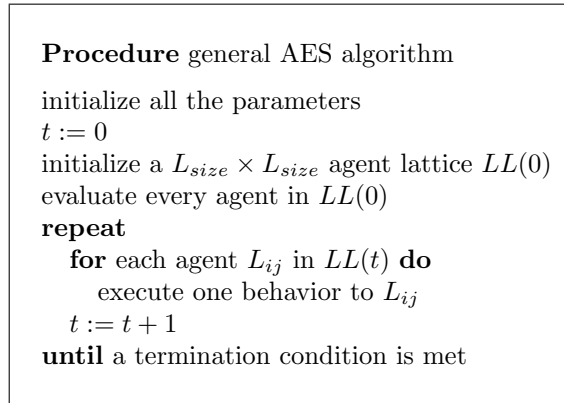


Figure 2: Pseudocode for general AES

## 2.2 Behaviors of Agents

In the proposed AES algorithm, all the agents update and co-evolve by executing a certain behavior. And the ideology of the behaviors is generated by imitating the character of local competitive and learning mechanism of the living organisms.

In nature, several organisms always work together to achieve a common goal. And many algorithms are formulated by simulating these natural phenomena.

In recent years, there has been an increasing concern from the evolution computation community on a class of hybrid EAs, named as memetic algorithms (MAs), which hybridizes local search (LS) methods to refine solution quality with EAs. The term MA is now widely used as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search. So far, MAs are extensively used for solving many optimization problems, such as scheduling problems [20], [1], travel salesman problems [27], combinatorial optimization problems [13], quadratic assignment problems [24] and other applications [6], [16].

Here, competitive and learning behaviors are defined. For each agent, if its energy is not the best, then it will execute the competitive behavior; if there's no better agent, the statistics-based learning behavior is executed by using the information feedback from the whole multi-agent system. The detailed discussion is given as follows.

### 2.2.1 Competitive Behavior

The apperceive range of each agent is 1, and the neighbor agents are called competitive neighborhood. Therefore, there are 8 agents in the competitive neighborhood of each agent. For an agent  $L_{i,j}$ , it will compare its energy with the agents in its neighborhood. If its energy is no less than any agent in its neighborhood, then it can live; else, it will extinct and its position will be replaced by the offspring of the best agent. The details are discussed further as follows.

Assume  $L_{i,j} = L_{i,j}^1, L_{i,j}^2, \dots, L_{i,j}^n, L_{max} = L_{max}^1, L_{max}^2, \dots, L_{max}^n$ , and  $\forall L \in N_{i,j}, E(L) < E(L_{max})$ . If  $E(L_{i,j}) < E(L_{max})$ , then  $L_{max}$  is used to generate an offspring  $L'_{i,j} = L'_{i,j}^1, L'_{i,j}^2, \dots, L'_{i,j}^n$  to replace  $L_{i,j}$ . There are two kinds of generating methods.

In the first competitive method [40], the information of  $L_{i,j}$  and  $L_{max}$  is utilized together to generate an offspring  $L'_{i,j}$ . It randomly chooses some position that  $L_{i,j}$  differs from  $L_{max}$  to alter the corresponding position in  $L_{i,j}$ .

$$L'_{i,j} = \begin{cases} L_{max}^k, & k \notin D \text{ or } (k \in D \text{ and } random(2) = 0) \\ 1 - L_{max}^k, & \text{otherwise,} \end{cases} \quad (3)$$

where  $k = 1, 2, \dots, n$ , set  $D$  record the sequence number of the position that  $L_{i,j}$  differs from  $L_{i,j}$ . Namely,  $D = \{k | L_{i,j}^k \neq L_{max}^k, k = 1, \dots, n\}$ .  $random(2)$  means to generate 0 or 1 randomly.

The second competitive method is the position based mutation, which is often used in evolutionary computation.

$$L'_{i,j} = \begin{cases} L_{max}^k, & rand() > \frac{1}{n} \\ 1 - L_{max}^k, & \text{otherwise,} \end{cases} \quad (4)$$

where  $k = 1, 2, \dots, n$ , and  $rand()$  means to generate a real number between 0.0 and 1.0.

The number of elements in set  $D$  actually equals to the Hamming distance between  $L_{i,j}$  and  $L_{max}$ . When  $|D|$  is small, it means the Hamming distance between  $L_{i,j}$  and  $L_{max}$  is small, and the two agents are similar. Then the probability of using the first competitive pattern to generate better agent is minor, thus we introduce a parameter  $D^k \in (0, 1)$  to determine which pattern should be used to generate  $L'_{i,j}$ : If  $|D|/n > D^k$ , then the first method is adopted; or else the second.

### 2.2.2 Statistics Based Learning Behavior

The purpose of the learning behavior is to enhance the energy of an agent by statistics based adaptive non-uniform mutation (SANUM) [35] or statistics based adaptive non-uniform crossover (SANUX) [34] with the *elite* (the best agent in the current generation). Because the resource is limited in the environment, only when the energy of an agent is no less than any of the agents in its neighborhood, it can get a chance to learn. The first learning scheme is statistics based mutation and the second is statistics based crossover with the *elite* agent. The schemes are described as follows.

First of all, we decide how to calculate the uniform crossover or mutation probability of a locus by using a statistics-based approach. A global information-based statistics, which has been proved to

be effective in enhancing the exploration capability of GAs by Yang [34, 35], is introduced here. Let  $p(i)$  denote the crossover or mutation probability of the locus  $i$ ,  $f_{ki}$  denote the frequency of  $k$ 's in the gene locus  $i$  over all the past generations, where  $i = 1, 2, \dots, n$  and  $k$  is the allele value for the gene locus. Then, in the binary encoding space, we have  $f_{1i} + f_{0i} = 1$ ,  $0 \leq f_{0i}, f_{1i} \leq 1$ , and  $f_{1i}$  can be regarded as the tendency to '1' for the locus  $i$  over all the past populations. SANUM and SANUX make use of this convergence information as feedback to control mutation and crossover by adjusting the probability of each locus. Thus the one-dimension statistic vector  $F1 = \{f_{11}, f_{12}, \dots, f_{1n}\}$  can express the convergence degree of the population from the gene level. Then,  $p(i)$  can be calculated from  $f_{1i}$  as follows:

$$p(i) = p_{max} - 2 \times |f_{1i} - 0.5| \times (p_{max} - p_{min}), \quad (5)$$

where  $|y|$  denotes the absolute value of  $y$ ,  $p_{min}$  and  $p_{max}$  denote the minimum and maximum allowable mutation or crossover probability for a locus respectively. For example, for the mutation probability, let  $p_{min} = 10^{-4}$  and  $p_{max} = \frac{1}{n}$ , and for the crossover probability, let  $p_{min} = 0.0$  and  $p_{max} = 0.5$ . Firstly, we calculate the distribution of 1's  $f_1(i)$  for each locus  $i$  over the lattice, and the crossover and mutation probability  $p(i)$  for that gene locus can be obtained. For the global statistics based method, the learning behavior can be adjusted by the convergence degree of the population.

The first learning scheme is the statistics based adaptive non-uniform mutation (SANUM). If  $L$  gets a chance to learn, then  $L$  is mutated to generate  $L'$ . If  $E(L') > E(L)$ , then agent  $L$  is replaced by  $L'$ .

Both SANUM and SANUX belong to the class of adaptive mechanisms that occur at the bottom level of mutation and crossover. Traditional bit mutation and crossover keep a constant probability over all the loci. In fact, as the population converges, fewer and fewer offsprings generated by mutating and crossover will survive in the next generation. That is, many operations are wasted on those converged loci. Most of these wasted operations and hence wasted fitness evaluations are saved by SANUM and SANUX through adaptively decreasing  $p(i)$  to  $p_{min}$  for those converged loci.

We introduce a population index  $\xi$  [26] in order to measure its diversity and it can be calculated as follows:

$$\xi = \frac{E_{best} - E_{ave}}{E_{best}}, \quad (6)$$

where  $E_{best}$  and  $E_{ave}$  are respectively the best and average energy among the energy values of the agents. Obviously, the index  $\xi$  can measure the state of agents via the energy calculation. When  $\xi$  decreases from 1 to 0, it means that the agents lose the diversity gradually.

The criterion of selecting a learning scheme is based on the value of  $\xi$  as follows.

- If  $\xi \leq 0.1$ , the first statistics based mutation learning scheme is applied considering the mutation operation can help a converging population to jump from a local optimum.
- If  $\xi \geq 0.9$ , the second statistics based crossover learning scheme is applied since the crossover operation can accelerate the exploitation to a diversifying population.
- If  $0.1 < \xi < 0.9$ , one of the two learning schemes is chosen randomly. *Random* means to generate a real number between 0 and 1. If  $rand() < 0.5$ , the first learning scheme is selected; else, the second one is chosen.

### 2.3 Two Schemes to Maintain the Diversity of Agents

In dynamic environments, the fitness landscape could change over time, that is, the current optimum point may become a local optimum and the past local optimum maybe become a new global optimum point. Since a spread-out population can adapt to these changes more easily, two diversity-keeping methods, namely random immigrants (RI) and adaptive dual mapping (ADM), will be introduced into our algorithm framework of AES algorithm to address DOPs.

### 2.3.1 Random Immigrants Method (RI)

Among the approaches developed for GAs to address DOPs, the random immigrants scheme has been proved to be a beneficial one for many DOPs. The random immigrants scheme aims to maintain the diversity of the population by replacing worst or randomly selected individuals from the population with randomly created individuals [41]. Here, we always choose the worst energy agents to be replaced by randomly generated individuals every generation. In order to avoid that random immigrants disrupt the ongoing search progress too much, especially during the period when the environment does not change, the ratio of the number of random immigrants to the population size is usually set to a small value, e.g., 0.1.

### 2.3.2 Adaptive Dual Mapping Method(ADM)

Dualism and complementarity are quite common in nature, such as the double-stranded structure in DNA molecules. Inspired by the complementarity mechanism in nature, a primal-dual genetic algorithm has been proposed and applied for DOPs [36]. Elitist is the currently-best solutions in subsequent generation. It preserves currently-best solutions from one generation to the next. The concept of elitism has been introduced into several EAs. Elitism has been considered an effective method for improving the efficiency of an EA[7][19][30]. Effective manipulation of the elite can contribute to better performance of EAs.

In this chapter, we investigate the application of the combination of dualism and elitism into AES. For the convenience of description, we first introduce the definition of dual agent here. Given an agent  $L$ , its dual agent is defined as  $L'$ , where  $L'^k = 1 - L^k, k = 1, \dots, n$ . With this definition, an agent *elite* is to evaluate its dual agent (*elite'*) firstly before executing statistics based learning. If its dual agent is evaluated to have more energy, that is,  $E(\text{elite}') > E(\text{elite})$ , then *elite* is replaced by *elite'*.

Given the above discussion, the proposed AES incorporate two diversity maintaining techniques (RI and ADM) is illustrated in Fig. 3.

## 3 The Dynamic Testing Suite

### 3.1 Stationary Test Problems

A set of well studied stationary problems, forming a range of difficulty levels for EAs, are selected as the experimental functions to compare the performance of AES with traditional standard GA (SGA), the Primal-Dual GA (PDGA) [36], and the GA with RI (RIGA), where the worst 10% individuals are replaced with random individuals every generation. In this chapter, dynamic test problems will be constructed from these stationary problems by a dynamic problem generator, which is called the XOR generator.

#### 3.1.1 One-Max Function

This problem simply aims to maximize ones in a binary string. The fitness of a string is the number of ones it contains. A string length of 100 bits will be used for this study. And the unique optimum solution has a fitness of 100.

#### 3.1.2 Royal Road Function

This function was proposed by Michel, Forrest and Holland [25]. It is defined on a 64 bit string consisting of eight contiguous building-blocks (BBs) and each BB has 8 adjacent bits. The fitness of this function is defined as follows:

$$f(x) = \sum_{i=1}^8 c_i \delta_i(x), \quad (7)$$

where  $c_i = 8$  and  $\delta_i = \{1, \text{if } x \in S; 0, \text{otherwise}\}$ . The optimal for this function is given as  $f(111\dots 1) = 64$ .

### 3.1.3 Deceptive Function

Deceptive functions are a family of functions where there exists low-order BBs that do not combine to form the higher-order BBs. In this study, a deceptive function is constructed, which consists of 10 copies of the order-4 fully deceptive function DF2 [33] and has an optimum fitness value of 300. DF2 is defined as follows:

f(0000)=28	f(0001)=26	f(0010)=24	f(0011)=18
f(0100)=22	f(0101)=6	f(0110)=14	f(0111)=0
f(1000)=20	f(1001)=12	f(1010)=10	f(1011)=2
f(1100)=8	f(1101)=4	f(1110)=6	f(1111)=30

**Procedure** proposed AES algorithm

initialize all the parameters

$t := 0$

initialize a  $L_{size} \times L_{size}$  agent lattice  $LL(0)$

evaluate every agent in  $LL(0)$

calculate crossover and mapping prob.  $p(i), i = 1, \dots, n$

**repeat**

**for** each agent  $L_{i,j}$  in  $LL(t)$  **do**

**if** there exists an agent  $L_{max} \in N_{i,j}$  with

$E(L_{max}) > E(L_{i,j})$  **then**

      calculate  $|D|/n$ , where  $|D|$  is the Hamming distance between  $L_{max}$  and  $L_{i,j}$

**if**  $|D|/n > D^h$  **then**

        select the first competitive scheme

**else** Select the second competitive scheme

**else**

$L'_{i,j} := \text{Dual}(L_{i,j})$

**if** ( $E(L'_{i,j}) > E(L_{i,j})$ ) **then**

$L_{i,j} := L'_{i,j}$

**if** ( $\xi < 0.1$ ) **then**

        select the first learning scheme to mutate  $L_{i,j}$

**else if** ( $\xi > 0.9$ ) **then**

        select the second learning scheme to crossover

$L_{i,j}$  with *elite*, which is the best agent so far

**else**

      randomly select the first or second learning scheme and execute it for  $L_{i,j}$

  calculate  $\xi$

$t := t + 1$

**until** a termination condition is met

Figure 3: Pseudocode for the proposed AES algorithm.

### 3.1.4 Double Deceptive Function

A six rank bipolar Double (Strongly Interrelated) Deceptive function [14] discussed here consists of 10 copies of 6 rank bipolar deceptive function DF1.

$$f(a) = \sum_{i=1}^{n/6} f_{bipolar6}(a_{6i-5}, a_{6i-4}, a_{6i-3}, a_{6i-2}, a_{6i-1}, a_{6i}) \quad (8)$$

DF1 is designed as follows:

$$f_{bipolar6}(a_1, a_2, a_3, a_4, a_5, a_6) = \begin{cases} 0.9, & \text{if } u = 3 \\ 0.8, & \text{if } u = 2 \text{ or } 4 \\ 0, & \text{if } u = 1 \text{ or } 5 \\ 1, & \text{if } u = 0 \text{ or } 6 \end{cases}$$

where  $u$  is the number of 1s in the variable. The strongly interrelated deceptive function has an optimum fitness of 10.

## 3.2 Generating Dynamic Test Problems

In this chapter, the dynamic test environments are constructed based on the above stationary functions by using an ‘‘XOR’’ operator [37, 39]. Suppose that the environment is periodically changed every  $\tau$  generations, the environmental dynamics can be formulated as follows:

$$f(x, t) = f(x \oplus M(k)), \quad (9)$$

where  $k = \lceil t/\tau \rceil$  is the period index,  $t$  is the generation counter, and  $M(k)$  is the XOR mask for period  $k$ . Given a value for parameter  $\rho$ ,  $M(k)$  can be incrementally generated as follows:

$$M(k) = M(k-1) \oplus T(k), \quad (10)$$

where  $T(k)$  is an intermediate binary template randomly created for period  $k$  containing  $\rho \times n$  ones. For the period  $k = 1$ ,  $M(1)$  is initialized to be a zero vector.

The advantage of this operator is the fitness landscape can still keep the certain properties in the original landscape after a change occurs, e.g., the total number of optima and fitness values of optima though their locations shifted. For example, if we apply a template  $T = 11111$  to a 5-bit One-Max function, the original optimal point  $x^* = 11111$  becomes the least fit point while the original worst point  $x = 00000$  becomes the new optimal point in the changed landscape, but the optimal fitness value (i.e., 5) and the uniqueness of optimum remain invariant.

In this chapter, we have constructed dynamic versions of above stationary problems. In our experiment, the change severity  $\rho$  is set to 0.1, 0.5 and 0.9, which is used to examine the algorithms’ performance in the changing environments with different severities respectively: from very slight change ( $\rho = 0.1$ ) to moderate variation ( $\rho = 0.5$ ) to intense change ( $\rho = 0.9$ ).

Another dynamics parameter considered is the change frequency, i.e. how often the environment changes. In our experiment the frequency parameter  $\tau$  is set to 20, 50 and 100 respectively because all algorithms are at quite early searching stage at generation 20, at medium searching stage at generation 50, and at late stage or converged at generation 100. By setting  $\tau$  to these values, we can inspect the capability of the algorithms adapting to dynamic environment under different level of searching stage.

Thus, we construct a series of 9 different dynamic problems from each stationary test problem. The dynamics parameter settings are summarized in Table 1.

## 4 Experimental Study

### 4.1 Experimental Setting

For AES algorithm,  $L_{size} = 10$ . When the mutation probability is computed,  $p_{min}$  is set to be  $10^{-4}$ ,  $p_{max}$  is  $1/n$ . And  $p_{min} = 0.0$  and  $p_{max} = 0.5$  are set for crossover probability computation.

Table 1: The index table for dynamic parameter setting.

$\tau$	Environmental Dynamics Index		
20	1	2	3
50	4	5	6
100	7	8	9
$\rho \rightarrow$	0.1	0.5	0.9

For SGA, RIGA and PDGA, parameters are set as follows: the population size *pop\_size* is set to 100, one-point crossover with a fixed probability  $p_c = 0.6$ , bit mutation with the mutation probability  $p_m = 0.001$ , and fitness proportionate selection with commonly used roulette wheel mechanism. The best  $N$  chromosomes among all the parents and children are always transferred to the next generation population. For every generation, the best-of-generation fitness was recorded. And for each run of an algorithm on a dynamic problem, 10 periods of environmental changes are allowed. Each experimental result is averaged over 100 runs with different random seeds.

For dynamic optimization problems, the optimal solution is not unique. Hence, it is not sufficient to compare only optimal solution obtained at last. It is extremely important to compare the performance of algorithms at every generation. The overall offline performance of an algorithm on a DOP is defined as:

$$\bar{F}_{BOG} = \frac{1}{G} \sum_{i=1}^G \left( \frac{1}{100} \sum_{j=1}^{100} \bar{F}_{BOG_{ij}} \right), \quad (11)$$

where  $G = 10\tau$  is the total number of generations for a run and  $\bar{F}_{BOG_{ij}}$  is the best-of-generation fitness of generation  $i$  of run  $j$ . The off-line performance  $\bar{F}_{BOG}$  is the best-of-generation fitness averaged over 100 runs and then averaged over the data gathering period.

## 4.2 Experimental Results on DOPs

The experimental results with respect to the mean best-of-generation fitness against evaluations of four EAs on the dynamic test problems are plotted in Fig. 4 to Fig.7 respectively.

The overall offline performance is presented in Fig.8. 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 2. In Table 2, the t-test result regarding Alg. 1 - Alg. 2 is shown as “s+”, “s-”, “+”, “-” or “~” when Alg. 1 is significantly better than, significantly worse than, insignificantly better than, insignificantly worse than, or statistically equivalent to Alg. 2 respectively. From these tables and figures several results can be observed.

First, when the environment shifted, AES can always outperform other peer EAs on almost all the problems, which has been indicated by the relevant *t*-test results in Table 2. AES can converge faster, and simultaneously it can trace the optimal solution better in dynamic environments. This result shows the efficiency of our proposed behaviors of the agents in dynamic environments. The competitive and learning behaviors of the agents, and the diversity maintaining schemes help the algorithm adapt well to the most changing environment. When the environment shifts slightly ( $\rho = 0.1$ ), AES significantly perform better over other EAs for the changing periods. Both PDGA and RIGA perform better than SGA, because the primal dual and RI operators can help them adapt to the new environments. But for One Max problem, when the environment changes slightly, the AES can be beaten by SGA and PDGA; When the shift severity is moderate ( $\rho = 0.5$ ), AES keeps performing perfectly well with the best-of-generation fitness staying in the optimum over all dynamic periods. SGA always maintains a lower fitness level than RIGA, PDGA, and AES can do for all environmental periods on all dynamic problems; when  $\rho = 0.9$ , both AES and PDGA perform much better than SGA and RIGA. It can

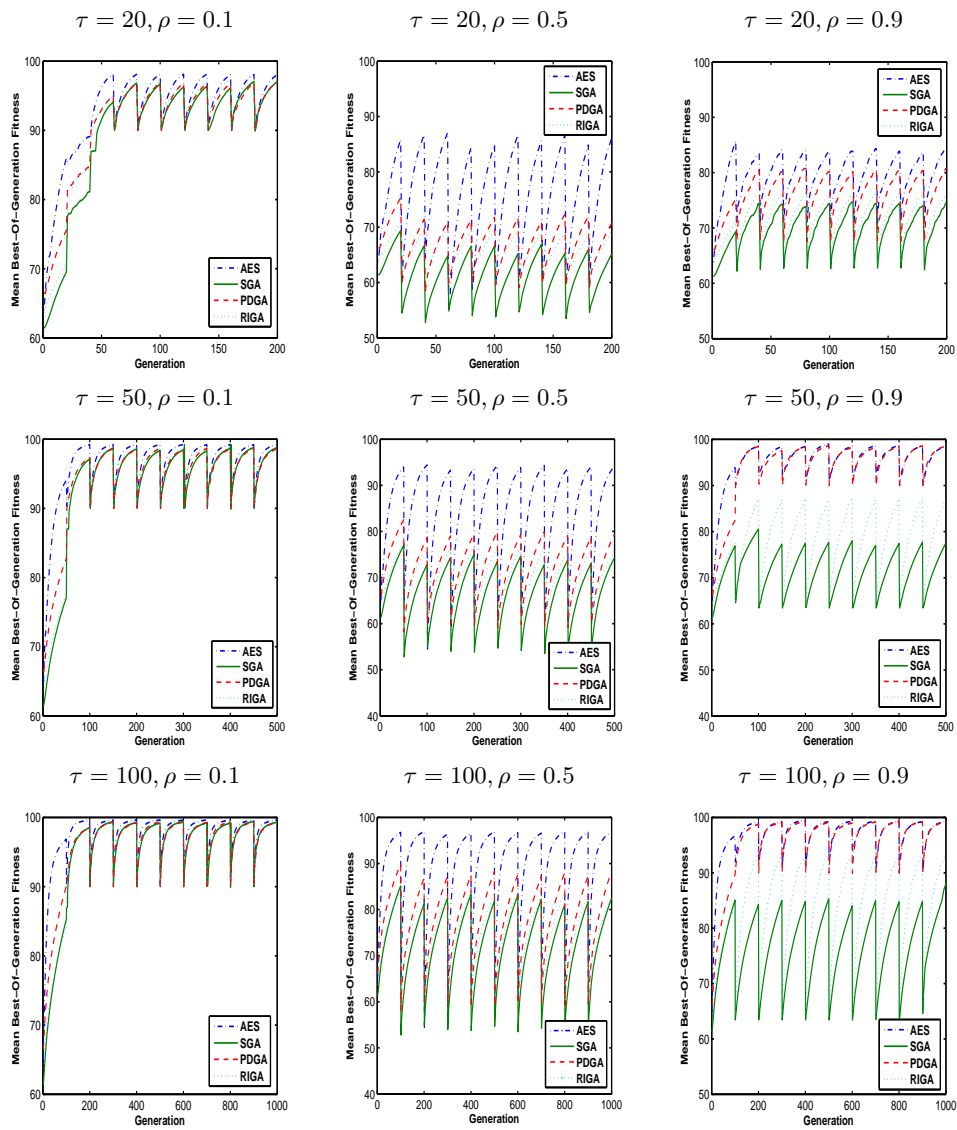


Figure 4: Dynamic performance of EAs on Dynamic One-Max Problems

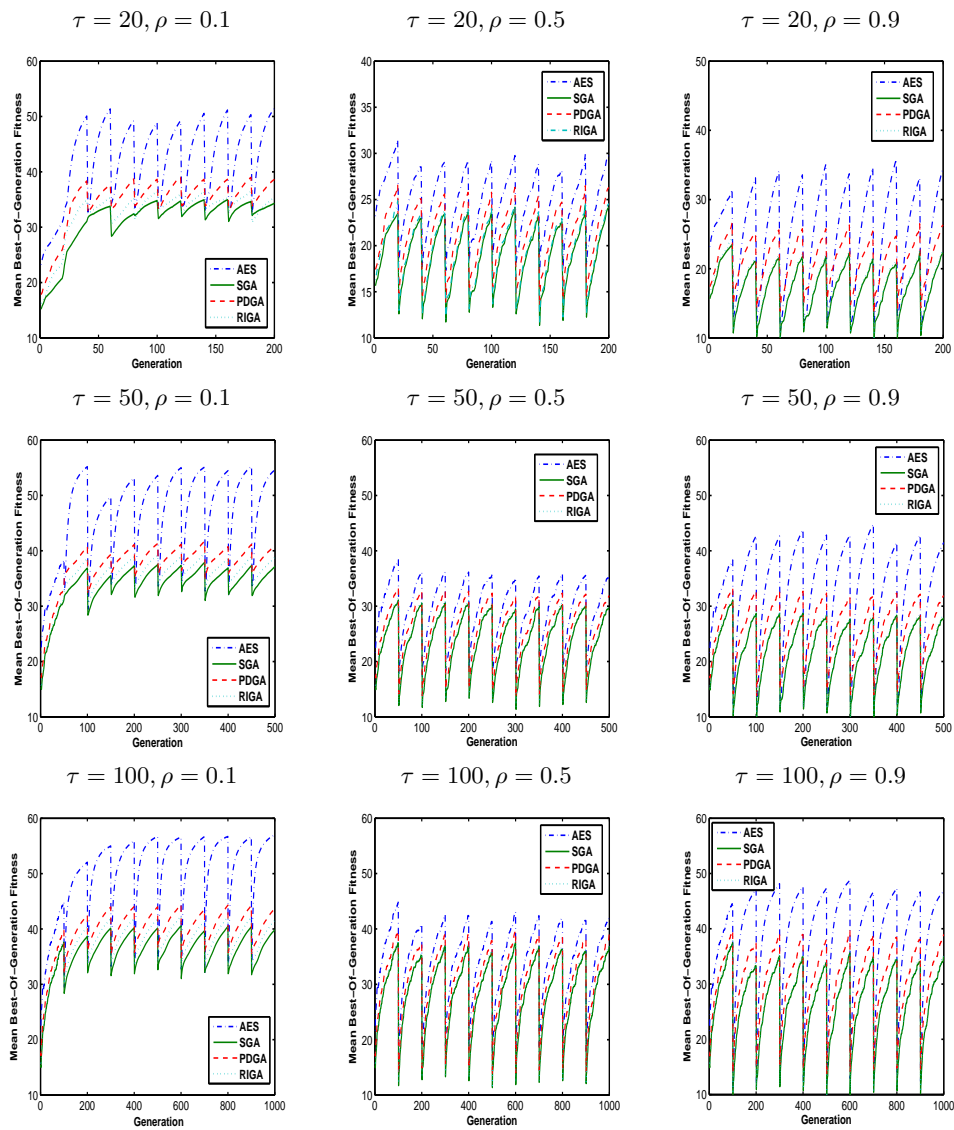


Figure 5: Dynamic performance of EAs on Dynamic Royal Road Problems

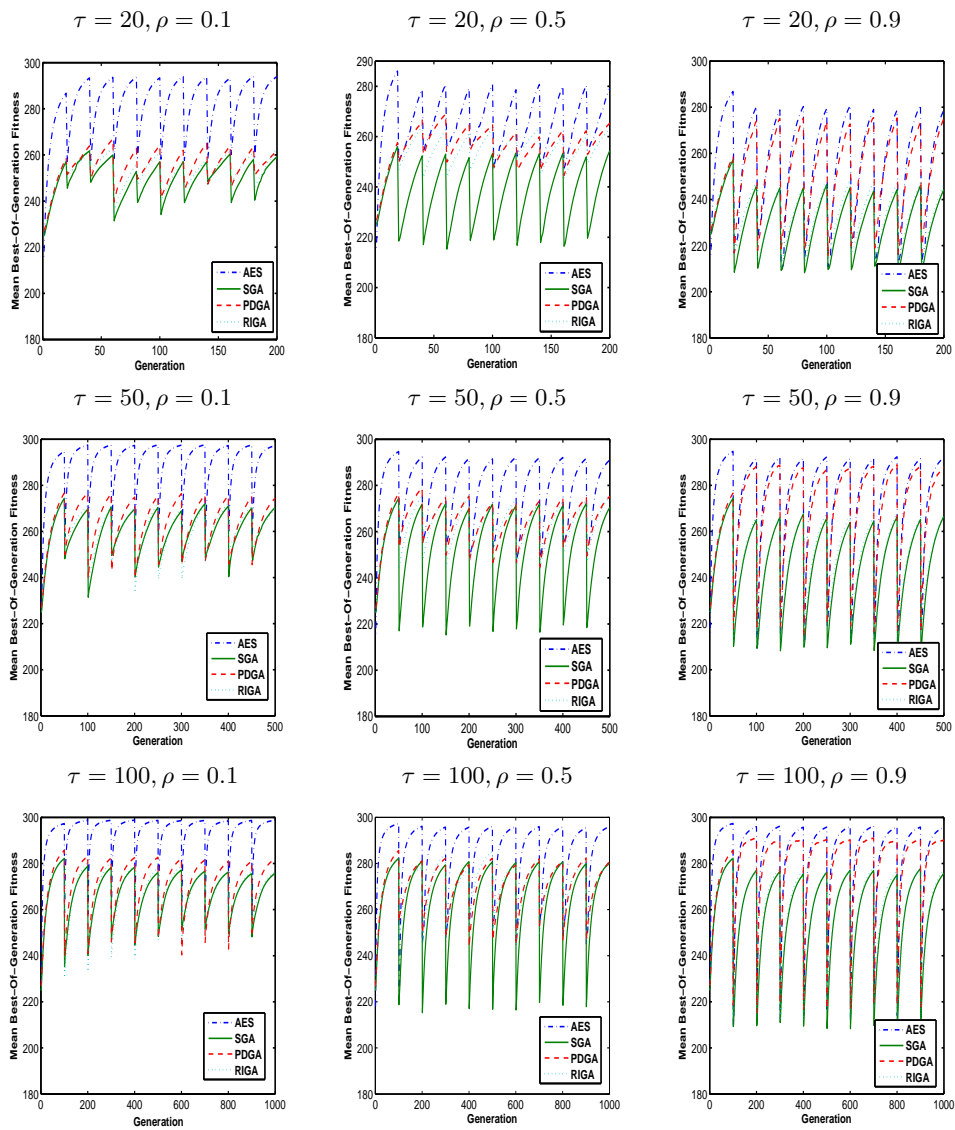


Figure 6: Dynamic performance of EAs on Dynamic Deceptive Problems

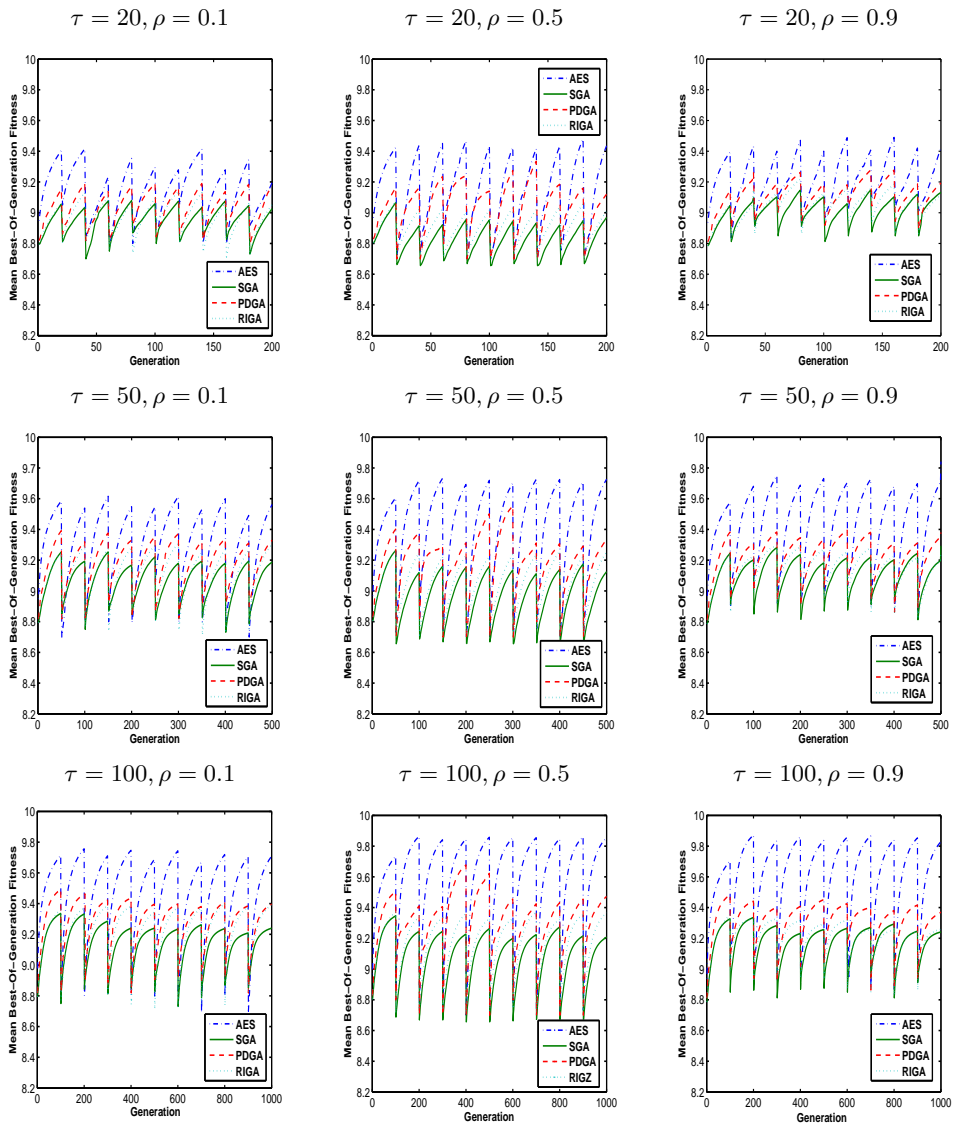


Figure 7: Dynamic performance of EAs on Dynamic Double Deceptive Problems

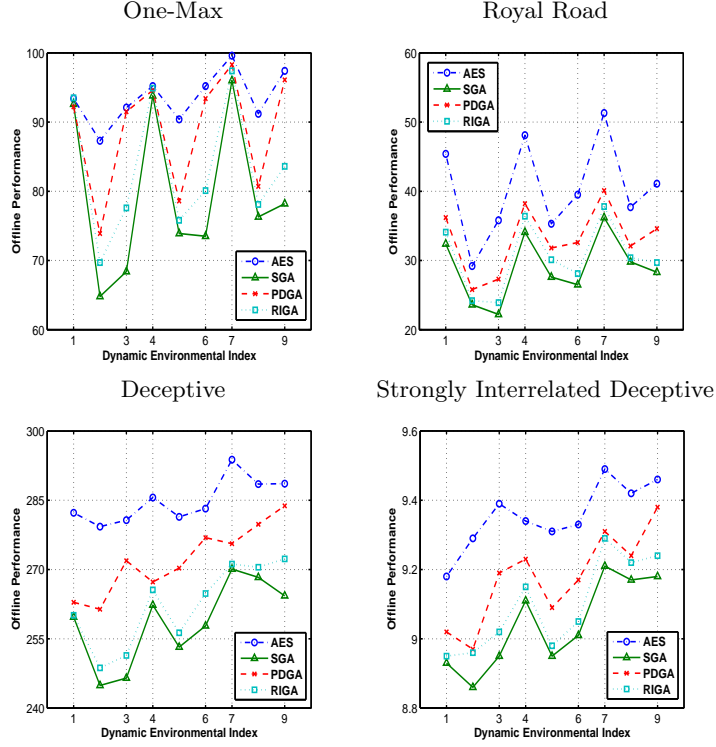


Figure 8: Overall Offline performance of EAs

Table 2: Statistical results of algorithms on dynamic problems (Alg.1–Alg. 4 denotes *AES*, *SGA*, *PDGA* and *RIGA* respectively).

<i>t</i> -test Result	One Max			Royal Road			Deceptive			Double Deceptive		
$\tau = 20, \rho \rightarrow$	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9
Alg.1-Alg.2	+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.1-Alg.3	~	s+	~	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.1-Alg.4	~	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.2-Alg.3	-	s-	s-	-	s-	s-	-	s-	s-	-	s-	s-
Alg.2-Alg.4	~	-	s-	-	~	-	~	-	-	~	s-	-
Alg.3-Alg.4	~	+	s+	+	+	s+	+	s+	s+	+	~	s+
$\tau = 50, \rho \rightarrow$	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9
Alg.1-Alg.2	+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.1-Alg.3	~	s+	~	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.1-Alg.4	~	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.2-Alg.3	-	s-	s-	-	s-	s-	-	s-	s-	-	s-	s-
Alg.2-Alg.4	-	-	s-	-	-	-	-	-	s-	-	-	-
Alg.3-Alg.4	~	+	s+	+	+	s+	+	s+	s+	+	s+	s+
$\tau = 100, \rho \rightarrow$	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9
Alg.1-Alg.2	+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.1-Alg.3	~	s+	~	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.1-Alg.4	~	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+	s+
Alg.2-Alg.3	-	s-	s-	-	s-	s-	-	s-	s-	s-	s-	s-
Alg.2-Alg.4	~	-	s-	-	~	-	~	-	s-	s-	-	-
Alg.3-Alg.4	~	+	s+	+	+	s+	+	s+	s+	~	~	s+

be shown from the figures that both AES and PDGA can converge fast and make good performance when intense shift happened.

Second, AES algorithm has a more quickly and robust converge capability than SGA, PDGA and RIGA. On the One-Max problem, when the environment shifts slightly, AES performs as well as SGA, PDGA and RIGA; when  $\rho = 0.5$ , AES keeps performing well with the best-of-generation fitness staying in the optimum over all dynamic periods; when  $\rho = 0.9$ , both AES and PDGA perform much better than SGA and RIGA, and also when change took place, it can be seen from the figures that both AESF and PDGA can converge fast and track the optimum when intense shift happens. On the Royal Road Problem and Deceptive Problem, when  $\rho = 0.1$  and  $\rho = 0.9$ , AES outperforms SGA, PDGA and RIGA; and when the environment half shift, both AES and PDGA perform well. The mean best-of-generation fitness first drops a little when changes occur, then rises up quickly to near optimum value. It is because the complementary and dominance mechanisms embedded in AES work perfectly under the condition of extreme environment change. On the the Strongly Interrelated Deceptive problem, AES outperforms SGA, PDGA and RIGA for all periods in all the shifting environments. For AES, whenever the fitness function changes, the statistics based mechanism and diversity maintaining mechanisms in AES rapidly draw back the best individuals in the last generation of previous period. And the lattice construction also helps to suit the new environment. And also, it can be seen from the figures, for all the DOPs, especially when the change is intense, that PDGA has the ability to track the optimal faster than SGA and RIGA. This lies in that dual mapping operator help PDGA adapt to the new environment quickly, maintain enough diversity, and therefore effectively track the new optimum after moderate and intense changes.

Third, the competitive and learning behaviors in AES algorithm make AES have a better exploitation ability than GAs. All the agents accomplish their update via competing in a local neighborhood or learning based on the statistical feedback information of the whole system, so AES algorithm has a rapid convergence capability.

Fourth, the combination of the two diversity maintaining schemes RI and ADM improves the adaptability of AES algorithm in non-stationary environments. The information of the former generation is reserved and the diversity is increased, therefore, AES algorithm with reserved information and multiplex diversity can adapt to the dynamic environment due to effectiveness of the combined maintaining scheme of RI and ADM.

Finally, the environmental parameters affect the performance of algorithms. The performance of all algorithms can be increased when the value of  $\tau$  arises from 20 to 50 to 100. It is easy to understand since algorithms have more time to find better solutions before the next change with the increment of the value of  $\tau$ . The effect of the changing severity parameter  $\rho$  is different. For example, if  $\rho$  is fixed, the performance curve of AES is always lower when the environment changes with a middle severity degree than that when  $\rho = 0.1$  and  $\rho = 0.9$ .

## 5 Conclusions

This chapter investigates the application of an agent-based evolutionary search algorithm (AES) for binary coded problems. Two special behaviors of agents, competing and learning, are proposed. Two diversity schemes, random immigrants and adaptive dual mapping scheme, are integrated into AES in order to improve its performance in dynamic environments. Based on the XOR dynamic problem generator, a series of dynamic test problems were constructed systematically, in the mean time, the proposed AES algorithm is used to solve the dynamic test problems.

From the experimental results, we can draw the following conclusions. First, the competitive and learning behaviors of agents can always help AES algorithm obtain a better performance than the peer GAs can do. Second, both random immigrants scheme and dual mapping method can help increase the population diversity of EAs. Combining them together also improves the performance of AES algorithm in dynamic environments efficiently. Third, some dynamic characteristics of the environments may affect the performance of algorithms.

Generally speaking, the experimental results indicate that the AES algorithm with the random

immigrants and adaptive dual mapping schemes exhibits better performance in dynamic environments. AES algorithm combining some diversity schemes seems to be a good choice to address DOPs.

The work studied in this chapter can be extended in several ways. It is interesting to hybridize AES algorithm with other advanced diversity schemes, e.g., hybrid memory schemes [32, 38]. Another interesting work is to further investigate the idea of AES algorithm to solve other optimization problems with sequential and real encodings in non-stationary environments.

## References

- [1] G. D. Annibale, R. D. Leone, P. Festa, and E. Marchitto. "A New Meta-Heuristic for the Bus Driver Scheduling Problem: GRASP Combined with Rollout," *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling*, pp. 192-197, 2007.
- [2] T. Blackwell, "Particle swarms and population diversity," *Soft Computing*, vol. 9, pp. 793-802, 2005.
- [3] T. Blackwell and J. Branke, "Multi swarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459-472, August 2006.
- [4] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, pp. 1875-1882, 1999.
- [5] J. Branke, T. Kaußler, C. Schmidh, and H. Schmeck, "A multi-population approach to dynamic optimization problems," *Proceedings of the 5th International Conference on Adaptive Computing in Design and Manufacturing*, pp. 299-308, 2000.
- [6] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A Fast Adaptive Memetic Algorithm for Online and Offline Control Design of PMSM Drives," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 37, no. 1, pp. 28-41, 2007.
- [7] W. A. Chang and R. S. Ramakrishna, "Elitism-Based Compact Genetic Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367-385, 2003.
- [8] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 523-530, 1993.
- [9] G. Cornujols, G. L. Nemhauser, and L. A. Wolsey, "The Uncapacitated Facility Location Problem," In P. B. Mirchandani and R. L. Francis (editors), *Discrete Location Theory*, Wiley NY, pp. 119-171, 1990.
- [10] P. Davidsson and F. Wernstedt, "A Multi-Agent System Architecture for Coordination of Just-in-time Production and Distribution," *Proceedings of the 17th ACM Symposium on Applied Computing*, ACM SAC, UK, pp. 294-300, 2002.
- [11] J. A. Persson and P. Davidsson, "Integrated optimization and multi-agent technology for combined production and transportation planning," *Proc. of the 38th Hawaii Int. Conf. on System Sciences*, IEEE Press, NY, pp. 1-9, 2005.
- [12] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligenc*, New York: Addison-Wesley, 1999.
- [13] J. E. Gallardo, C. Cotta, and A. J. Fernandez, "On the Hybridization of Memetic Algorithms with Branch-and-Bound Techniques," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 37, no. 1, pp. 77-83, 2007.
- [14] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithm revisited: studies in mixed size and scale," *Complex Systems*, vol. 4, no. 4, pp. 145-444, 1990.

- [15] J. Grefenstette, "Genetic algorithms for changing environments," *Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature*, pp. 137-144, 1992.
- [16] R. J. W. Hodgson, "Memetic Algorithms and the Molecular Geometry Optimization Problem," *Proceedings Of the 2000 Congress on Evolutionary Computation*, pp. 625-632, 2000.
- [17] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous Agents and Multi-Agent Systems Journal*, vol. 6, no. 4, pp. 317-331, 1998.
- [18] A. Karageorgos, N. Mehandjiev, G. Weichhart, and A. Hammerle, "Agent-based optimization of logistics and production planning," *Engineering Application of Artificial Intelligence*, vol. 16, pp. 335-348, 2003.
- [19] J. L. Kim and R. D. Ellis Jr, "Permutation-Based Elitist Genetic Algorithm for Optimization of Large-Sized Resource-Constrained Project Scheduling," *Journal of Construction Engineering and Management*, vol. 134, no. 11, pp. 904-913, 2008.
- [20] B. Liu, L. Wang, and Y. H. Jin, "An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 37, no. 1, pp. 18-27, 2007.
- [21] J. Liu, "Autonomous Agents and Multi-Agent Systems," *Explorations in Learning Self-Organization, and Adaptive Computation*, Singapore: World Scientific, 2001.
- [22] J. Liu, Y. Y. Tang, and Y. C. Cao, "An evolutionary autonomous agents approach to image feature extraction," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 141-158, 1997.
- [23] S. S. Manvi and M. N. Birje, "An agent-based resource allocation model for grid computing," *Proceedings of 2005 IEEE International Conference on Services Computing*, IEEE Press, NY, pp. 311-314, 2005.
- [24] P. Merz and B. Freisleben, "A Comparison of Memetic Algorithms, Tabu Search and Ant Colonies for the Quadratic Assignment Problem," *Proceedings Of the 1999 Congress on Evolutionary Computation*, 2063-2070, 1999.
- [25] M. Mitchell, S. Forest, and J. H. Holland, "The royal road for genetic algorithms: fitness landscape and GA performance," *Proceedings of the 1st European Conference on Artificial Life*, pp. 245-254, 1992.
- [26] F. Neri, J. Toivanen, and R. A. E. Makinen, "An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV," *Applied Intelligence*, vol. 27, no. 3, pp. 219-235, 2007.
- [27] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 37, no. 1, pp. 92-99, 2007.
- [28] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [29] M. Sun, "Solving the uncapacitated facility location problem using tabu search," *Computers and Operations Research*, vol. 33, no. 9, pp. 2563-2589, 2006.
- [30] G. Tan, D. Zhou, and B. Jiang, "MI Dioubate Elitism-based immune genetic algorithm and its application to optimization of complex multi-modal functions," *Journal of Central South University of Technology*, vol. 15, no. 6, pp. 845-852, 2008.
- [31] H. Wang and D. Wang, "An improved primal-dual genetic algorithm for optimization in dynamic environments," *Proceedings of the 13th International Conference on Neural Information Processing*, LNCS 4234, vol. 3, pp. 836-844, 2006.

- [32] H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," *Applications of Evolutionary Computing*, LNCS 4448, pp. 637-646, 2007.
- [33] L. D. Whitley, "Fundamental principles of deception in genetic search," *Foundations of Genetic Algorithms I*, pp. 221-241, 1991.
- [34] S. Yang, "Adaptive non-uniform crossover based on statistics for genetic algorithms," *Proceedings of the 2002 Genetic and Evolutionary Computation Conference*, pp. 650-657, 2002.
- [35] S. Yang, "Adaptive mutation using statistics mechanism for genetic algorithms," In F. Coenen, A. Preece, and A. Macintosh (editors), *Research and Development in Intelligent Systems XX*, London: Springer-Verlag, pp. 19-32, 2003.
- [36] S. Yang, "PDGA: the primal-dual genetic algorithm," In A. Abraham, M. Koppen, and K. Franke (editors), *Design and Application of Hybrid Intelligent Systems*, Sydney: IOS Press, pp. 214-223, 2003.
- [37] S. Yang, Non-stationary problem optimization using the primal-dual genetic algorithm. *Proceedings of the 2003 Congress on Evolutionary Computation*, vol. 3, pp. 2246-2253, 2003.
- [38] S. Yang, "Memory-based immigrants for genetic algorithms in dynamic environments," *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, vol. 2, pp. 1115-1122, 2005.
- [39] 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.
- [40] W. C. Zhong, J. Liu, M. Z. Xue, and L. C. Jiao, "A multiagent genetic algorithm for global numerical optimization," *IEEE Transactions on System, Man, and Cybernetics-Part B*, vol. 34, no. 2, pp. 1128-1141, 2004.
- [41] F. Vavak and T. C. Fogarty, "A comparative study of steady state and generational genetic algorithms for use in nonstationary environments," In T. C. Fogarty (editors), *DAISB Workshop on Evolutionary Computing*, LNCS 1443, pp. 297-304, 1996.