ACADEMIC World Academic Union

ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 5, No. 4, 2010, *pp*. 287-298

Cellular Genetic Algorithm with Density Dependence for Dynamic Optimization Problems

Hao Chen [1], Ming Li [2, +] and Xi Chen [2]

[1] College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

[2] Key Laboratory of Nondestructive Test, Nanchang Hangkong University, Nanchang 330063, China

(*Received October 15, 2010, accepted October 19, 2010*)

Abstract. For dynamic optimization problems, the aim of an effective optimization algorithm is both to find the optimal solutions and to track the optima over time. In this paper, we advanced two kinds of cellular genetic algorithms inspired by the density dependence scheme in ecological system to solving dynamic optimization problems. Two kinds of improved evolution rules are proposed to replace the rule in regular cellular genetic algorithm, in which null cells are considered to the foods of individuals in population and the maximum of living individuals in cellular space is limited by their food. Moreover, in the second proposed rule, the competition scheme of the best individuals within the neighborhoods of one individual is also introduced. The performance of proposed cellular genetic algorithms is examined under three dynamic optimization problems with different change severities. The computation results indicate that new algorithms demonstrate their superiority respectively on both convergence and diversity.

Keywords: cellular genetic algorithm, dynamic optimization, density dependence scheme

1. Introduction

Most of the optimization problems in real world are dynamic optimization problems (DOPs). Evolutionary algorithms (EAs) have been widely and successfully applied to solve static optimization problems (SOPs). However, the evaluation function, design variables, and the constraints are not fixed in DOPs. Hence, for DOPs the aim of an effective optimization problem is not only to find the optimal solution but also to track the optima over time.

In recent years, there is a growing interest in studying evolutionary algorithms (EAs) for DOPs, and several approaches have been developed, such as increasing diversity after a change via hyper mutation [1] or random immigrants[2], maintaining diversity throughout the run [3,4], memory schemes [5,6], and multi-population approaches [7,8].

Cellular genetic algorithm (CGA) is a subclass of genetic algorithms (GAs); it is set up through an organic combination of evolutionary computation and cellular automata. In CGA, the population is arranged in a given grid, the evolution of each individual is restricted in its neighborhood, and each individual is only allowed to genetic operate with the individuals in its neighborhood. With the distributed arrangement, CGA has a good performance on maintaining genetic diversity which is important to find and approximate the dynamic optimum for DOPs. Hence, CGA is considered to be a significant and meaningful algorithm to solving DOPs.

The research on combining ideas from cellular automata with genetic algorithms began in Manderick and Spiessens's work [9]. Over the past decade or so, CGAs have been proven to be effective for solving many kinds of optimization problems from both classical and real world settings.

Many kinds of improved CGAs were proposed for optimizations. Kirley [10] introduced a novel evolutionary algorithm named cellular genetic algorithm with disturbances inspired by the nature of spatial interactions in ecological systems. Simoncini et al. [11] presented an anisotropic selection scheme for CGA, improved the performance by enhance diversity and control the selective pressure. Janson and Alba [12]

+ Corresponding author. Tel.: +86-0791-386 3695; fax: +86-0791-386 3695.
E-mail address: limingniat@hotmail.com

proposed a hierarchical CGA, where the population structure was augmented with a hierarchy according to the fitness of individuals. Nebro et al [13] introduced an external archive in CGA to store the better solutions, the search experience contained in the archive were feed backed into algorithm though replacement strategy. Ishibuchi et al. [14] proposed a new CGA with two neighborhood structures: one for global elitism, the other for local competition among neighbors.

Besides, the theoretical research of CGAs is also active. Giacobini et al. [15] presented a theoretical study of the selection pressure in asynchronous CGAs with different evolution rules. Alba et al. [16] presented a comparative study of several asynchronous policies for updating the population in CGAs. Zhang [17] researched the evolution rules of optimization algorithm with cellular automata from the ability of life reproduction and the probability of survival.

In this paper we investigate an improved cellular genetic algorithm to solving DOPs. Inspired by density dependence scheme in the nature, we propose a new evolution rules. The paper is structured as follows. Section 2 reviews some related work on CGA. Section 3 introduces the regular CGA with evolution rules. In Section 4, two density dependence schemes are introduced, population control scheme is also discussed, and a cellular genetic algorithm with density dependence is proposed. Section 5 introduces the DOPs chosen and presents the results of proposed algorithm. Section 6 briefly expressed the conclusion of this paper.

2. Cellular Genetic Algorithm with Evolution Rules

2.1. Basic concepts

A cellular automaton can be denoted as $\boldsymbol{A} = (L_d, S, N_d, f)$ mathematically, in which $\boldsymbol{A}$ is a cellular automaton; L_d is the cellular space; S is the set of states of cell, each cell only has one state such as “living” or “dead”; N_d denotes the neighborhood of a cell such as Von. Neumann-type, Moore-type, Ex-Moore-type; f is the local transfer function which defines the state of the center cell by the states of its neighbors, and can be called evolution rule.

Fig.1 shows the Moore-type in grid, in which a cell in the small black square is the center cell; cells within two squares are the neighborhood of the center cell; the grey means living, contrarily, the white means dead. In this paper, the proposed algorithm uses this type.

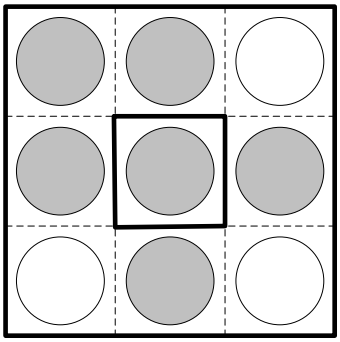


Fig. 1: Moore-type in regular grid

2.2. Cellular genetic algorithm with evolution rule

The pseudo-code of cellular genetic algorithm with evolution rule is shown in Fig.2. In this algorithm, each living individual only interacts and genetic operates with individuals in its neighborhood. The fitness value of offspring individuals will be calculated, if an offspring is better than the center cell individual, the old center one will be replaced during the next generation. After the genetic operation, state of each individual will be update by the evolution rule.

```
begin
  t=0
  initialize population P(0) and individuals' states
  repeat
      for each living individual I_i
         denote the neighborhood of I_i by N_i
         select the best individual in N_i, denote by I_i'
          //normal genetic operation
          offspring= crossover([I_i I_i'],p_c)
          offspring= mutate(offspring,p_m)
          f_offspring=evaluate(offspring)
          if min(f_offspring)<f(I_i) then
            replace I_i with the best individual in offspring
      end for
      update the state of each individual by the evolution rule
   until terminated=true
end
```

Fig. 2: Pseudo-code of cellular genetic algorithm with evolution rule

In CGA, the complex optimization problems can be solved by some simple rules. In order to simulate the biological evolution more effectively, it is important to introduce evolution rule of 'living' or 'dead' state of cell. In the next loop, the state of a cell depends on the states of its neighbors. The game of life evolution rule is a typical evolution rule. The mathematical formula is shown as follow:

$$\text{Rule:} \quad \begin{aligned} &\text{if } S^t = 1 \text{ then } S^{t+1} = \begin{cases} 1 & |N_S| = 2,3 \\ 0 & |N_S| \neq 2,3 \end{cases} \\ &\text{if } S^t = 0 \text{ then } S^{t+1} = \begin{cases} 1 & |N_S| = 3 \\ 0 & |N_S| \neq 3 \end{cases} \end{aligned} \tag{1}$$

3. Proposed Algorithms

Among the existing research, most of evolution rules in CGAs are directly introduced from cellular automaton. For these evolution rules, although complex system behavior can be obtained by simple settings, but the interaction between individuals and the relationship between evolution scheme and group behavior of individuals had been ignored.

In nature, the state ("living" or "dead") of individual in population depends on the structure of living space, the mortality rate and survival rate are dependent by the density of population in living environment. When the density is lower, the food is adequate for the population, and the survival rate increases. When the density is higher, food supplies are scarce, intraspecific competition becomes seriously and the mortality rate increases.

Inspired by this actual phenomenon, two kinds of local density dependence schemes are introduced in this section and cellular genetic algorithms with density dependence are also proposed.

3.1. Local density dependence scheme within neighborhood

The null cells in grid space are considered as the food of individuals in population, an individual in a cell means the food in this cell is occupied by the individual.

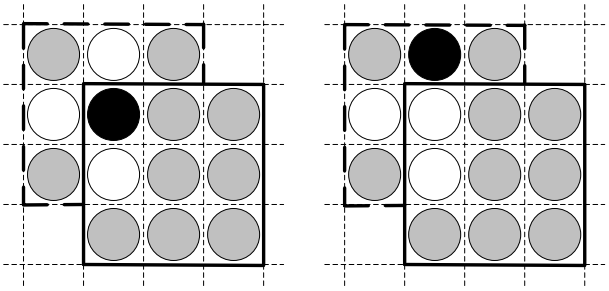
Definition 1: Living density. The ratio between the number of living individuals and the number of food in a region is considered to the living density in this region.

Specially, the living density in the neighborhood of an individual x_i is called the local living density of the individual which can be denoted as $LD(x_i)$. Taking Moore-type for instance, the local living density of the center individual in Fig.1 is 0.667.

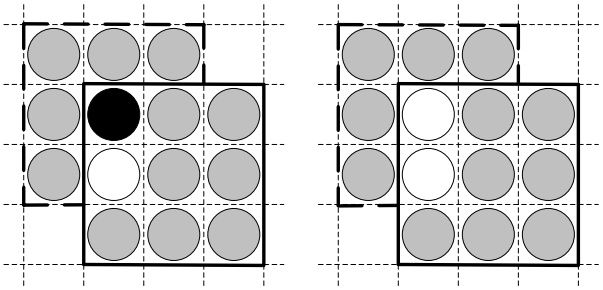
Definition 2: Maximum local living density. In a living space, foods for the population are limited. Hence, the local density is limited in the scope of the food supplied, the upper limit of the scope is considered as maximum local living density which is denoted as LD_{Max}.

In neighborhood of an individual, if the local living density is more than LD_{Max}, the surplus weakest

individuals will dead or escaped from the region because of the shortage of food. The density dependence scheme in neighborhood of an individual is shown in Fig.3 (LD_{Max} is set to 0.8). Same as the setting shown in Fig.1, the grey means living, and the white means dead. The region within solid line is the current living space; the black cell in it means the surplus weakest individual; the region within dashed box is the escape area of the surplus individuals.



a. the surplus individual escape from a region



b. the surplus individual dead

Fig. 3: Density dependence operation

With the effect of density dependence, if there is more than one null cell in the escape area as shown in left figure of Fig.3a, the surplus individual will escape from the former region; contrarily, if there is no null cell in the escape area, the surplus individual will dead as shown in right figure of Fig.3b. Additionally, if a surplus individual had been escaped from a region, it is not permitted escape twice; in other words, it will dead.

The detail algorithm of local density scheme within neighborhood can be described in Fig.4. After the density dependence operation, the structure of the population and the states of cells corresponding to the individuals will change. In the later sections, this operation is named density dependence I, is denoted as $[P'(t), \boldsymbol{S}'(t)]$=*Dependence*-I $(P(t), \boldsymbol{S}(t))$ in which $\boldsymbol{S}(t)$ means the states of whole cells in grid space.

Besides local density dependence as previously described, the intraspecific competition also contains the competition between the best individuals within a living space. For genetic algorithm and some similar algorithms, premature convergence is one of the main factors restricted the optimization performance, and the best individual is replicated many times due to the effect of select operation. This is the main reason to this appearance. Hence, we introduced an adding operation into the density dependence scheme to avoid two or more same structure individuals appeared in the neighbourhood of an individual. The detail setting is also described in Fig.4, and in the later sections, we call this operation by density dependence II, and denote it by $[P'(t),\boldsymbol{S}'(t)]$=*Dependence*-II $(P(t),\boldsymbol{S}(t))$.

```
denote the current population of living individuals by P_L(t)
ranking P_L(t) by fitness from better to worse: Index=ranking(P_L(t))
for i=1:Index
  denote the current individual and its neighborhood by I_i and N_i
  for each living individuals in N_i // for density dependence II
      calculate the hamming distance to I_i, denote as HD
    if HD<1
      change the state of the cell to 'dead'
  end
  denote the number of living individuals in N_i by n_i
  if (n_i+1)/N_d>LD_Max //density dependence available
    for each surplus weakest individual I_j'
      denote the escape area of I_j' by N_j'
      change the state of the current cell of I_j' to 'dead'
      if I_j' has not been escaped from a region
        if there is null cell in N_j'
          select a random null cell
          change the individual in the cell by I_j'
          change the state of the cell to 'living'
    end for
end for
```

Fig. 4: Pseudo-code of density dependence I, II

3.2. Population control scheme

The density dependence scheme is acting among the individuals in same generation. Due to the separate effect of density dependence, the number of living individuals after operation is less than or equals to the number before operation. In this section, the scheme of population control is defined.

In nature, the population growth is relevant to the abundance of food, and cannot grow unlimited. In grid space, the anticipant number of living individuals in next generation is related to the number of null cells in current generation. We define a mathematical formula as follow to determine the number of living individuals in next generation.

$$N_{t+1} = N_t + N_t(1-\frac{N_t}{N\cdot\alpha}) \tag{2}$$

where N_t and N_{t+1} are the number of living individuals in t and t+1 generation, N is the number of cells in cellular space, α is a rate which control the maximum number of individuals feed, $\alpha \in [LD_{\max},1]$, especially when $\alpha = 1$ means the food in one cell can feed one individual. In the later sections, we denote this operation by $|P(t+1)|$=population-control($|P'(t)|$).

3.3. Cellular genetic algorithm with density dependence

We introduce the local density dependence scheme and population control scheme to the cellular genetic algorithm and propose a new cellular genetic algorithm. The pseudo-code of the new cellular genetic algorithm is shown in Fig.5.

```
begin
 t=0
 initialize population P(0) and individuals' states
 repeat
  //genetic operation
    for each living individual I_i
       denote the neighborhood of I_i by N_i
       select the best individual in N_i, denote by I_i'
       offspring= crossover([I_i I_i'],p_c)
       offspring= mutate(offspring,p_m)
       f_offspring=evaluate(offspring)
       if min(f_offspring)<f(I_i) then
         replace I_i with the best individual in offspring
    end for
    //for density dependence
    [P'(t), S'(t)]=Dependence-I (P(t), S(t))
    // for density dependence II
    [P'(t), S'(t)]=Dependence-II (P(t), S(t))
    //population control
     |P(t+1)|=population-control(|P'(t)|)
     //determine P(t+1) and S(t+1)
    if |P(t+1)|>|P'(t)| then
     P(t+1):=placed |P(t+1)|-|P'(t)| random individuals in null cells
     S(t+1):=change the state of corresponding cell to 'living'
    else
        P(t+1):=delete |P'(t)|-|P(t+1)| worst individuals in P'(t)
    S(t+1):=change the state of corresponding cell to 'dead'
  until terminated=true
end
```

Fig. 5: Pseudo-code of cellular GA with density dependence I, II

4. Experimental Results

4.1. Test problems

Yang's DOP generator [18] can construct random dynamic environments from any binary-encoded stationary function by an exclusive OR operator. For a static optimization problem $f(x)$, create intermediate binary template $\boldsymbol{T}=[T_1,T_2,\ldots T_N]$ by a designated method, N is the number of change. The expression of the DOP in ith environment is shown as follow:

$$F_i(x)=f(x\oplus T_i) \tag{3}$$

where $\oplus$ denotes the XOR operator. The severity of environmental changes is determined by the percentage of 1 in the template $\boldsymbol{T}$ which is denoted as s, the frequency is controlled by the generation interval between two adjacent changes which is denoted as τ, the complexity is affected by the periodically structure of $\boldsymbol{T}$.

Three 100-bit binary functions, denoted One-Max, NK(25, 4) and Deceptive respectively, are selected as base stationary functions to construct DOPs. Construct test DOPs from these stationary functions by Yang's DOP generator, where s is set to 0.1, 0.3, 0.5, and 0.9, τ is set to 25, $\boldsymbol{T}$ is set to random, cyclical and cyclical with noise [6]. The detail describe of there DOPs are shown respectively as follow.

In static One-Max problem, the fitness value of individual is assigned by the number of the same bits between individual and the given template. The mathematical formula of the dynamic One-Max is shown as follow:

$$F_i(x)=f(x\oplus T_i)=L-\sum((x\oplus T_i)\oplus B) \tag{4}$$

where B is the given template, $L=100$ is the length of binary code string. It has an optimum fitness of 100 in each environment.

Static NK(25,4) problem consist of 25 contiguous 4-bit building blocks $\{s_k\},k=1,\cdots,25$, the calculation formula is

$$f(x)=\sum_{k=1}^{25} c_k \delta_k(x) \tag{5}$$

where $c_k=4$, $\delta_k(x)=\begin{cases}1, & x\in s_k\\ 0, & \textbf{\textit{else}}\end{cases}$. The dynamic NK(25,4) is shown as follow:

$$F_i(x)=f(x\oplus T_i)=\sum_{k=1}^{25} c_k \delta_k(x\oplus T_i) \tag{6}$$

Same to the dynamic One-Max, it has an optimum fitness of 100 in each environment.

Static deceptive problem is a fully deceptive problem, it also consist of 25 contiguous 4-bit building blocks $\{s_k\}, k=1,\cdots,25$, the calculation formula is

$$f(x)=\sum_{k=1}^{25} d(H(x_k,s_k)) \tag{7}$$

where x_k is the corresponding sub-string with s_k in individual x, $H(x_k,s_k)$ denotes the hamming distance between x_k and s_k, $d(*)$ is a mapping function, $d(*)$ is 4,0,1,2,3 respectively when $*$ is 0,1,2,3,4. The dynamic deceptive problem is shown as follow:

$$f(x)=\sum_{k=1}^{25} d(H((x\oplus T_i)_k,s_k)) \tag{8}$$

It also has an optimum fitness of 100 in each environment.

4.2. Results and discussion

Three kinds of the dynamic test optimization problems are optimized respectively by four algorithms: cellular genetic algorithm with density dependence I (CGA-DI), cellular genetic algorithm with density dependence II (CGA-DII), cellular genetic algorithm with evolution rule (CGA-R) and improved simple genetic algorithm (ISGA). Experiments were carried out to compare the performance of algorithms on the dynamic test environments. For all algorithms, the parameters are set as follows: population size N=100, max generations G=500, uniform crossover operator with crossover rate P_c=0.7, discrete mutation operator with rate P_m=0.01.

The experimental results of convergence performances of CGA-DI, CGA-DII, CGA-R and ISGA are summarized in Tables I, II, III with the form of average $\pm$ the standard error. The convergence metric is used to measure the overall convergence performance of algorithms, which is defined as

$$F_{acc}=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{K}\sum_{j=1}^{K}F_{ij} \tag{11}$$

where n=20 is the total number of runs, $K=G/\tau$ is the total number of environmental changes, F_{ij} is the optimum of jth change in ith run. Table I, II, III are the results of random, cyclical and cyclical with noise DOPS respectively. Fig.6 shows the diversity behavior of algorithms in random dynamic environment, Fig.6a shows the results on dynamic One-Max problem, Fig.6b shows the results on dynamic NK(25,4) problem, Fig.6c shows the results on dynamic deceptive problem. The diversity metric measures the extent of diversity achieved among the individuals, which is defined as

$$\gamma=\frac{1}{N}\sum_{j=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left(-\sum_{k=1}^{s_c}P_{lk}\log(P_{lk})\right) \tag{12}$$

where N is the population size, L is the length of chromosome, s_c is the cardinality of genotypic alleles, P_{lk} is the rate of kth genotypic allele appear on lth location, the maximum of γ is ln2.

Table I. Experimental results on random DOPs

	s	CGA-DI	CGA-DII	CGA-R	ISGA
One-Max	0.1	**99.99 ± 0.02**	99.35 ± 0.77	97.87 ± 2.20	83.85 ± 1.40
	0.3	**99.72 ± 0.06**	96.10 ± 0.35	87.35 ± 0.37	68.79 ± 3.13
	0.5	**99.79 ± 0.11**	95.29 ± 0.35	82.02 ± 1.50	63.61 ± 3.88
	0.9	**99.79 ± 0.09**	95.26 ± 0.41	79.88 ± 1.97	59.97 ± 5.65
NK(25,4)	0.1	77.14 ± 3.30	**83.76 ± 2.78**	70.34 ± 3.37	43.89 ± 4.07
	0.3	63.01 ± 2.09	**71.88 ± 1.47**	48.71 ± 4.00	27.56 ± 6.30
	0.5	62.59 ± 2.41	**71.25 ± 1.30**	41.02 ± 5.25	24.23 ± 6.27
	0.9	59.71 ± 3.77	**70.21 ± 1.79**	42.81 ± 4.59	29.19 ± 6.84
Deceptive	0.1	82.81 ± 2.19	**84.47 ± 0.70**	79.37 ± 1.82	64.09 ± 1.63
	0.3	81.40 ± 1.08	**83.11 ± 0.46**	72.65 ± 1.35	55.17 ± 4.14
	0.5	80.98 ± 0.68	**83.73 ± 0.56**	71.56 ± 1.47	53.80 ± 4.07
	0.9	84.58 ± 1.27	**86.73 ± 0.77**	82.67 ± 1.89	65.56 ± 1.74

Table II. Experimental results on cyclical DOPs

	s	CGA-DI	CGA-DII	CGA-R	ISGA
One-Max	0.1	**99.99 ± 0.02**	98.57 ± 0.73	97.77 ± 0.63	93.32 ± 4.16
	0.3	**99.82 ± 0.10**	95.71 ± 0.34	90.23 ± 1.03	83.85 ± 5.31
	0.5	**99.82 ± 0.06**	95.41 ± 0.39	87.45 ± 1.21	80.42 ± 6.62
	0.9	**99.99 ± 0.02**	98.78 ± 0.79	97.82 ± 1.22	93.57 ± 4.03
NK(25,4)	0.1	79.49 ± 6.11	**85.99 ± 3.03**	81.64 ± 2.78	61.66 ± 4.90
	0.3	68.98 ± 2.30	**74.36 ± 1.60**	59.50 ± 3.24	51.55 ± 11.55
	0.5	64.04 ± 1.74	**71.41 ± 1.42**	54.86 ± 4.34	48.43 ± 14.50
	0.9	80.39 ± 5.05	**86.02 ± 2.19**	84.95 ± 2.20	62.25 ± 5.02
Deceptive	0.1	82.58 ± 0.83	**87.51 ± 0.55**	78.84 ± 0.43	74.60 ± 2.91
	0.3	83.20 ± 0.86	**86.30 ± 0.64**	74.74 ± 1.28	69.19 ± 4.80
	0.5	83.64 ± 0.39	**85.96 ± 0.73**	74.36 ± 1.12	68.85 ± 5.17
	0.9	81.79 ± 0.79	**87.92 ± 0.65**	81.50 ± 0.58	76.07 ± 2.43

Table III. Experimental results on cyclical with noise DOPs

	s	CGA-DI	CGA-DII	CGA-R	ISGA
One-Max	0.1	**99.97 ± 0.05**	98.49 ± 0.66	97.20 ± 0.68	92.69 ± 4.34
	0.3	**99.75 ± 0.10**	95.66 ± 0.31	89.09 ± 0.99	81.06 ± 4.98
	0.5	**99.82 ± 0.07**	95.25 ± 0.27	86.42 ± 1.41	79.93 ± 5.75
	0.9	**99.97 ± 0.04**	98.58 ± 0.59	97.44 ± 0.91	92.62 ± 4.18
NK(25,4)	0.1	79.35 ± 5.19	**83.95 ± 2.17**	79.64 ± 2.33	61.46 ± 4.20
	0.3	68.93 ± 2.34	**73.58 ± 1.42**	57.88 ± 3.86	50.00 ± 10.22
	0.5	63.73 ± 2.45	**71.41 ± 1.58**	53.12 ± 3.76	47.48 ± 13.17
	0.9	79.10 ± 6.09	**85.10 ± 2.93**	82.83 ± 2.21	61.98 ± 5.44
Deceptive	0.1	81.73 ± 0.85	**86.07 ± 0.38**	77.99 ± 0.65	73.92 ± 2.82
	0.3	82.25 ± 0.65	**85.16 ± 0.56**	72.67 ± 1.23	67.77 ± 5.74
	0.5	83.11 ± 0.60	**83.90 ± 0.55**	73.35 ± 0.93	68.59 ± 4.07
	0.9	81.63 ± 1.15	**87.11 ± 0.51**	80.21 ± 0.35	74.42 ± 2.39

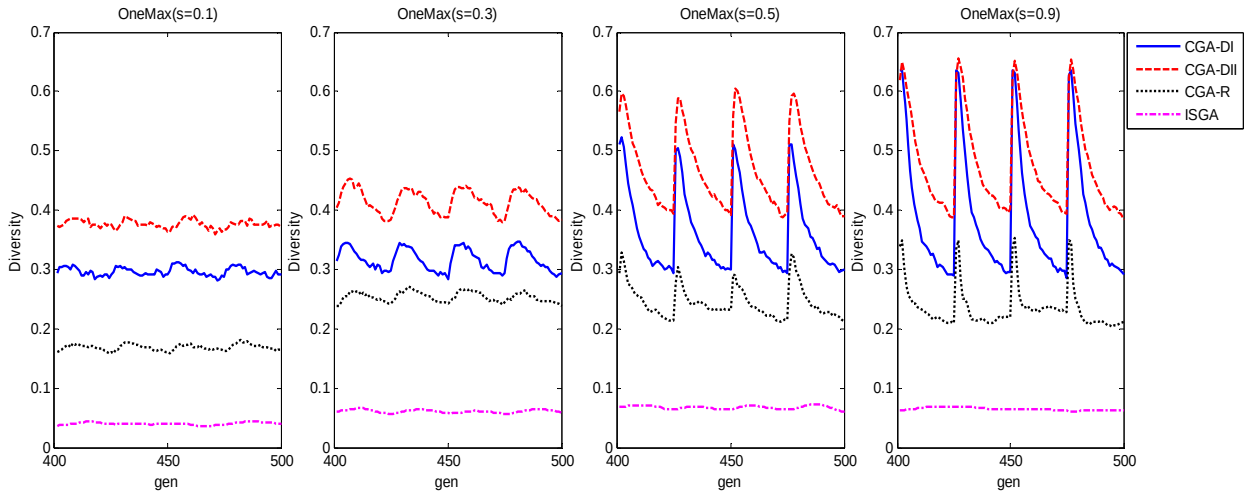
1) The convergence performance of algorithms in dynamic environments

From Tables I, II, III, it is interesting to note that CGA-DI can obtain the best results in dynamic One-Max problems with different severity and complexity among four algorithms; CGA-DII has a good performance in dynamic NK(25,4) and deceptive problems can obtain the best results in dynamic One-Max problems with different severity and complexity; ISGA is the worst one. Moreover, for all algorithms, they get the best performance in dynamic One-Max problems, and get the worst performance in dynamic NK(25,4)

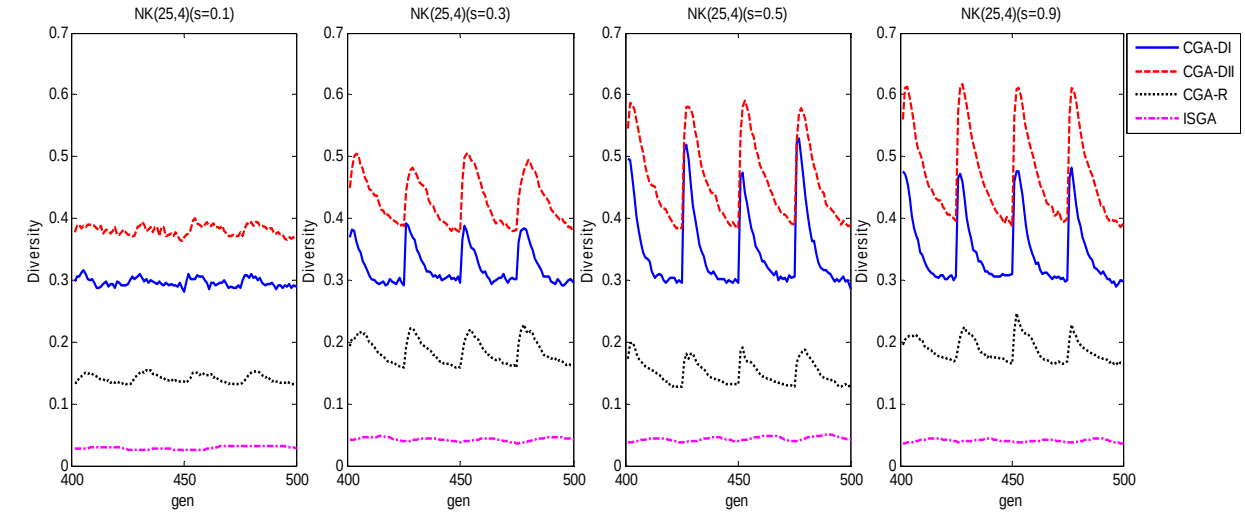
problems. The detail discussions are shown as follow:

In random dynamic One-Max problem, CGA-DI has the best convergence performance, the relative errors of F_{acc} with global optimal solution under 4 different change severities all lower than 0.5%; for CGA-DII, CGA-R and ISGA, the F_{acc} reduced with the increase of s, the relative errors of F_{acc} get minimums of 0.65%, 2.13% and 16.15% respectively when s=0.1, 0.65%, and get maximums of 4.74%, 20.12% and 40.03% respectively when s=0.9. In random dynamic NK(25,4) problem, for CGA-DI and CGA-DII the F_{acc} reduced with the increase of s, the relative errors of F_{acc} t get minimums of 22.86% and 16.24% when s=0.1, and get maximums of 40.29% and 19.79% when s=0.9; for CGA-R and ISGA the F_{acc} reduced and then increased with the increase of s, the relative errors of F_{acc} get minimums of 29.66% and 56.11% when s=0.1, and get maximums of 58.98% and 75.77% when s=0.5. In random dynamic deceptive problem, for all algorithms the F_{acc} reduced and then increased with the increase of s, the relative errors of F_{acc} get minimums of 15.42%, 13.27%, 17.33% and 34.44% respectively when s=0.9, and get maximums of 9.12%, 16.37%, 28.44% and 46.20% respectively when s=0.5.

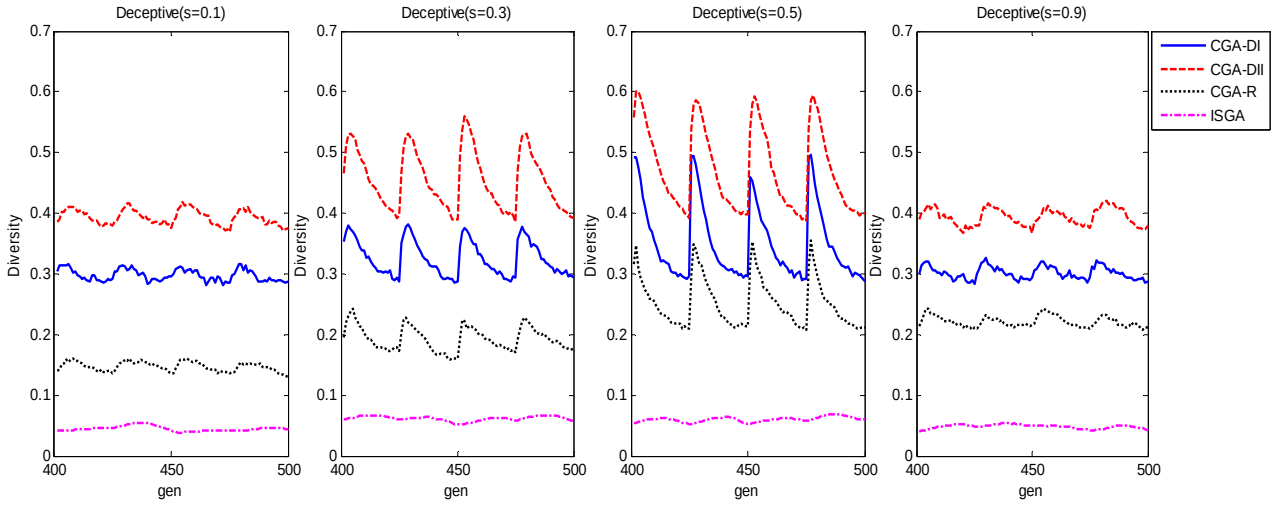
The mutative trends of F_{acc} of algorithms for cyclical and cyclical with noise DOPs are similar, and F_{acc} is reduced with the introduction of noise. In cyclical dynamic One-Max problem, for all algorithms the F_{acc} reduced and then increased with the increase of s, the relative errors of F_{acc} get minimums of 0.01%, 1.22%, 2.18% and 6.43% respectively when s=0.9, and get maximums of 0.18%, 4.59%, 12.55% and 19.58% respectively when s=0.5; with the introduction of noise, the largest percentage decrease with 0.07%, 0.20%, 1.26% and 3.33% respectively. In cyclical dynamic NK(25,4) problem, for all algorithms the F_{acc} reduced and then increased with the increase of s, the relative errors of F_{acc} get minimums of 19.61%, 13.98%, 15.05% and 37.75% respectively when s=0.9, and get maximums of 35.96%, 18.59%, 45.14% and 51.57% respectively when s=0.5; with the introduction of noise, the largest percentage decrease with 1.60%, 2.37%, 3.17% and 3.01% respectively. In cyclical dynamic deceptive problem, for CGA-DII, CGA-R and ISGA, the F_{acc} reduced and then increased with the increase of s, the relative errors of F_{acc} get minimums of 12.08%, 18.50% and 23.93% respectively when s=0.9, and get maximums of 14.04%, 25.64% and 31.15% respectively when s=0.5; for CGA-DI, the F_{acc} increased and then reduced with the increase of s, the relative errors of F_{acc} get a minimum of 16.56% when s=0.5, and get a maximum of 18.21% when s=0.9; with the introduction of noise, the largest percentage decrease with 1.14%, 2.40%, 2.40% and 2.17% respectively.



a. dynamic One-Max problem



b. dynamic NK(25,4) problem



c. dynamic Deceptive problem

Fig. 6 Diversity behavior of algorithms in random dynamic environments

2) The diversity performance of algorithms in dynamic environments

From Fig.6, it is interesting to note that all the curves of diversity metric change cyclically with the change of environments; the diversity curve of CGA-DII is better than that of the other algorithm; the diversity curve of ISGA is worse than that of the others. The detail discussions are shown as follow:

In random dynamic One-max problems, the variation within two adjacent changes of diversity curves and the diversity levels of algorithms increased with the increase of s. For CGA-DI, the diversity curve is smooth and near 0.3 when s=0.1, and changes sharply between 0.3 and 0.65 when s=0.9; for CGA-DII, the diversity curve is smooth and near 0.4 when s=0.1, and changes sharply between 0.4 and 0.65 when s=0.9; for CGA-R, the diversity curve is locate between 0.1 and 0.2 when s=0.1, and has a cyclical change between 0.2 and 0.35 when s=0.9; for ISGA, the diversity curve is locate below 0.1 in each value of s.

In random dynamic NK(25,4) problems, the trend of diversity curves is similar to that in One-Max problems. Differently, the variation within two adjacent changes of diversity curves becomes smoother than that in One-Max problems and the diversity levels of algorithms becomes lower than that in One-Max problems, such as the diversity curve of CGA-DI changes between 0.3 and 0.5 when s=0.9; for CGA-R, the diversity curve has a change near 0.2 when s=0.9

In random deceptive problems, the variation of diversity curves increased and then reduced with the increase of s. The change of diversity curves when s=0.1 is similar to that when s=0.9. The severity of

diversity curves get the most violent when s=0.5.

5. Conclusion

In this study, two kinds of evolution rules with density dependence for cellular genetic algorithm are discussed, and the corresponding cellular genetic algorithms with density dependence are proposed. Compared with regular cellular genetic algorithm with evolution rule, new algorithms can obtain superior convergence and diversity performance. According to the experiments carried on the dynamic test problems selected, CGA-DI can obtain the best results in dynamic One-Max problems and CGA-DII has a good performance in dynamic NK(25,4) and deceptive problems.

6. Acknowledgements

This work is supported by National Natural Science Foundation (NNSF) of China (No. 60963002), the Natural Science Foundation of Jiangxi Province of China (No. 2009GZS0090), and the Open Fund of the Key Laboratory of Nondestructive Testing, Ministry of Education, Nanchang Hangkong University (No. ZD200929006).

7. References

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

[2] J. J. Grefenstette. Genetic algorithms for changing environments. in *Proc. of the 2nd Int. Conf. on Parallel Problem Solving from Nature*. 1992, pp. 137-144.

[3] K. W. Yeom, J. H. Park. Biologically inspired evolutionary agent systems in dynamic environments. in *Proc. of the 2006 Congress on Evolutionary Computation*. Vancouver. 2006, pp. 386-390.

[4] M. Maury, J. Gouvea, F. R. Aluizio. Diversity-based model reference for genetic algorithms in dynamic environment. in *Proc. 2007 Congress on Evolutionary Computing.* Singapore. 2007, pp. 4639-4645.

[5] A. Simoes, E. Costa. Variable-size memory evolutionary algorithm to deal with dynamic environments. M. Giacobini et al. Eds. *LNCS 4448*. Springer-Verlag. 2007, pp. 617-626.

[6] S. Yang, X. Yao. Population-based incremental learning with associative memory for dynamic environments. in *Proc. of Congress on Evolutionary Computing*. Hong Kong. 2008, pp. 1-20.

[7] J. Branke, T. Kaubler, C. Schmidt. A multi-population approach to dynamic optimization problems. in *Adaptive Computing in Design and Manufacturing*. Berlin. 2000, pp. 299-308.

[8] T. Blackwell, J. Branke. Multi-swarm optimization in dynamic environments. *LNCS 3005*. Springer-Verlag. 2007, pp. 489-500.

[9] B. Manderick, P. Spiessens. Fine-grained parallel genetic algorithms. J. D. Schaffer Eds. in *Proc. of the third international conference on genetic algorithms*. San Mateo: Morgan Kaufmann. 1989, pp. 428-433.

[10] M. Kirley. A cellular genetic algorithm with disturbances: optimisation using dynamic spatial interactions. *Journal of Heuristics*. 2002, **8**(3): 321-342.

[11] D. Simoncini, P. Collard, S. Vérel, M. Clergue. From cells to islands: an unified model of cellular parallel genetic algorithms. in *Proc. 7th International Conference on Cellular Automata, for Research and Industry*. Perpignan, France. 2006, pp. 248-257.

[12] S. Janson, E. Alba, B. Dorronsoro, M. Middendorf. Hierarchical cellular genetic algorithm. in *Proc. of 6th European Conference on Evolutionary Computation in Combinatorial Optimization*. Budapest, Hungary. 2006, pp. 111-122.

[13] A. Nebro, J. Durillo, F. Luna, B. Dorronsor, E. Alb. A MOCell: a cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*. 2009, **24**(7): 726-746.

[14] H. Ishibuchi, N. Tsukamoto, Y. Nojima. Use of local ranking in cellular genetic algorithms with two neighborhood structures. in *Proc. of 7th International Conference on Simulated Evolution and Learning.* Australia: Melbourne. 2008, pp. 309-318.

[15] M. Giacobini, E. Alba, M. Tomassini. Selection intensity in asynchronous cellular evolutionary algorithms. in *Proc. 2003 International Conference on Genetic and Evolutionary Computation*. Chicago. 2003, pp. 955-966.

[16] E. Alba, K. Doerner, B. Dorronsoro. Adapting the savings based ant system for non-stationary vehicle routing problems. in *Proc. of First Conference on Metaheuristics and Nature Inspired Computing.* Tunisia, Hammamet. 2006.

[17] Y. Zhang, M. Li, Y. Lu. Study on evolution rules of optimization genetic algorithm with cellular automata.

Application Research of Computers. 2009, 26(10): 3635-3638.

[18] S. Yang. Constructing dynamic test environments for genetic algorithms based on problem difficulty. in *Proc. of the 2004 Congress on Evolutionary Computation*. Seattle. 2004, pp. 1262-1269.