アルゴリズム 69－6
（1999．9．2）

多様性制御指向遺伝的アルゴリズム(DCGA)：関数の最適化における性能

下平丞作士
文教大学　情報学部　情報システム学科

概要： 遺伝的アルゴリズムの探索過程において、解が局所的最適解に落ち込まないで真の最適解に到達するためには、集団の遺伝子型の適切な多様性を維持する必要がある。このような目標を達成するため、DCGA (Diversity-Control-oriented Genetic Algorithm)と呼ぶ新しい遺伝的アルゴリズムを提案した。DCGA では、次世代の個体は、親と生成した子供を合わせた集団から、最良の適応度の個体と候補の個体間のハミング距離に基づいた選択確率を用いて選択される。既報では、採用している方法の有効性を示す実験結果について報告した。本報告では、DCGA の関数最適化における性能を、ベンチマーク問題についての実験により検討している。行った実験の範囲では、DCGA は優れた性能を示し、既往の研究で提案されている有力な手法と比肩しうるものと考えられる。

A Diversity-Control-Oriented Genetic Algorithm (DCGA): Performance in Function Optimization

Hisashi Shimodaira
Department of Information and Communication, Bunkyo University

Abstract: In genetic algorithms, in order to attain the global optimum without getting stuck at a local optimum, an appropriate diversity of structures in the population needs to be maintained. I have proposed a new genetic algorithm called DCGA (Diversity-Control-oriented Genetic Algorithm) to attain this goal. In DCGA, the structures of the population in the next generation are selected from the merged population of parents and their offspring on the basis of a selection probability, which is calculated by using a hamming distance between a candidate structure and the structure with the best fitness value. In the previous papers, the effectiveness of DCGA has been shown by some experiments. In this paper, the performance of DCGA in function optimization is examined by experiments on benchmark problems. Within the range of my experiments, DCGA showed superior performances and it seems to be a promising competitor of previously proposed algorithms.

1. Introduction

Genetic algorithms (GAs) are a promising means for function optimization. One problem plaguing traditional genetic algorithms is convergence to a local optimum. The genetic search process converges when the structures in the population are identical, or nearly so. Once this occurs, the crossover operator ceases to produce new structures, and the population stops evolving. Unfortunately, this often occurs before the true global optimum has been found. This behavior is called *premature convergence*. The intuitive reason for premature convergence is that the structures in the population are too alike. This realization suggests that one method for preventing premature convergence is to ensure that the different members of the population are different, that is, to maintain the diversity of structures in the population [1].

Various methods for maintaining the diversity of structures in the population have been proposed to improve the performance of genetic algorithms. These are classified into two groups: methods for the selection process that can select and keep different structures in the population and those for genetic operators (mutation and crossover) that can produce different offspring. Major ones of the former are Mauldin's method [1] in which a new structure is checked whether it differs from every other structure in the population on the basis of the Hamming distance, Goldberg's sharing function method [2] in which a potential fitness is modified by the sharing function value that is calculated on the basis of the distance between each structure, and Mahfoud's deterministic crowding method [3] in which the selection is performed between two parents and their offspring on the based of the phenotypic similarity measure. Major ones of the latter are Eshelman's CHC [4, 5] that employs a highly disruptive uniform crossover (HUX) and the population-elitist selection (PES) method, Srinivas 's AGA [6] in which the probabilities of crossover and mutation are varied depending on the fitness value of the structure, Shimodaira's GALME [7, 8] that employs a large mutation rate controlled by a decreasing function of generation and the PES method, Sefrioui's distance-

dependent mutation [9] whose rate is changed dynamically depending on the distance between the individuals that are crossed, and Yang's CGA [10] that uses family competition and decreasing-based Gaussian mutation. Many studies have shown that the performance of CHC on standard benchmark tests is good and robust [11, 12].

From the results of these previous studies, it turns out that maintaining an appropriate diversity of the structures in the population is effective for avoiding premature convergence and for improving the performance, whereas the proposed methods are not necessarily complete. Therefore, I have developed a new genetic algorithm called DCGA (Diversity-Control-oriented Genetic Algorithm) [13, 14, 15, 16, 17]. In DCGA, the structures in the next generation are selected from the merged population of parents and their offspring with duplicates eliminated on the basis of a selection probability, which is calculated by using the hamming distance between a candidate structure and the structure with the best fitness value. The selection probability is larger for a structure with a larger hamming distance. The diversity of structures in the population can be externally controlled by adjusting the coefficients of the probability function so as to be in an appropriate condition according to the given problem. Within the range of some experiments, DCGA outperformed the simple GA and seems to be a promising competitor of the previously proposed algorithms.

This paper describes the outline of DCGA and presents the results of experiments to examine the performance of DCGA in function optimization. The results are compared with those for the promising previous studies.

2. The outline of DCGA

2.1 Algorithm

The skeleton of DCGA is shown in Fig. 1. The number of structures in the population P(t) is constant and denoted by N, where t is the generation number. The population is initialized by using uniform random numbers. In the selection for reproduction, select$_r$, all the structures in P(t-1) are paired by selecting randomly two structures without replacement to form P'(t-1). That is, P'(t-1) consists of N/2 pairs. By applying mutation with probability p_m and always applying crossover to the structures of each pair in P'(t-1), C(t) is produced. The mutation rate p_m is constant for all the structures. The structures in C(t) and P(t-1) are merged and sorted in order of their fitness values to form M(t). In the selection for survival, select$_s$, those structures that include the structure with the best fitness value are selected from M(t) and the population in the next generation P(t) is formed.

The details of the selection for survival, select$_s$ are as follows:

```
begin
  t=0;
  initialize population P(t);
  evaluate structures in P(t);
  while (termination condition not satisfied) do;
  begin;
    t=t+1;
    select_r P'(t-1) from P(t-1) by randomly pairing
      all structures without replacement;
    apply mutation with p_m and crossover to each
      pair of P'(t-1) and form C(t);
    evaluate structures in C(t);
    merge structures in C(t) and P(t-1) and sort
      them in order of their fitness values to
      form M(t);
    select_s N structures including the structure with
      the best fitness value from M(t) to form the
      next population P(t) according to the
      following procedure:
      (1) eliminate duplicate structures in M(t) to
          form M'(t);
      (2) select structures from M'(t) with CPSS
          method;
      (3) if the number of selected structures is
          smaller than N, introduce new structures
          by the difference of the numbers;
  end;
end;
```

Fig. 1. Skeleton of DCGA.

① Duplicate structures in M(t) are eliminated and M'(t) is formed. Duplicate structures mean that they have identical entire structures.

② Structures are selected by using the Cross-generational Probabilistic Survival Selection (CPSS) method, and P(t) is formed from the structure with the best fitness value in M'(t) and the selected structures. In the CPSS method, structures are selected by using uniform random numbers on the basis of a selection probability defined by the following equation:

$$p_s = \{(1-c)h/L + c\}^{\alpha}, \qquad (1)$$

where h is the hamming distance between a candidate structure and the structure with the best fitness value, L is the length of the entire string representing the structure, c is the shape coefficient, and α is the exponent. In the selection process, a uniform random number in the range [0.0, 1.0] is generated for each structure. If the generated random number is smaller than p_s that is calculated by Eq.(1) for the structure, then the structure is selected; otherwise, it is deleted. The selection process is performed in order of the fitness values of all the structures in M'(t),

without considering the fitness value of a structure itself, except the structure with the best fitness value.

③ If the number of the structures selected in the process ② is smaller than N, then new structures generated by using uniform random numbers are introduced by the difference of the numbers.

2.2 Empirical and Theoretical Justification

The reasons for employing the above methods in DCGA are as follows.

Side-effects of crossover and mutation may destroy the better-performing structures obtained so far. In DCGA, because the structure with best performance obtained so far always survives intact into the next generation, the influence of this side-effect is small. Therefore, large mutation rates can be used and crossover can always be applied. This results in producing offspring that are as different as possible from their parents and in examining regions of the search space that have not yet been explored. In fact, the best result was obtained when a crossover rate of 1.0 was used. On the other hand, in the simple GA, mutation is a background operator, ensuring that the crossover has full range alleles so that the adaptive plan is not trapped on a local optimum [18], and low mutation rates are generally used.

Duplicate structures reduce the diversity of the structures in the population and often cause premature convergence, because the same structures can produce a large number of offspring with the same structure in the next generation. Therefore it is effective to eliminate duplicate structures in order to avoid premature convergence.

Preliminary experiments on functions for the selection probability showed that the performance is closely related to the shape of the curve. Eq. (1) was selected because it can easily and flexibly express various curves. Example curves of Eq. (1) are shown in Fig. 2.

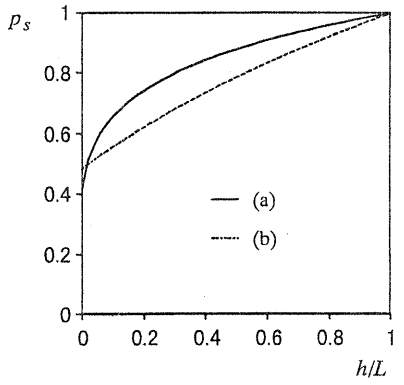


Fig.2. Example curves of Eq. (1). (a) $\alpha = 0.19$, $c = 0.01$; (b) $\alpha = 0.5$, $c = 0.234$.

The structure with the best fitness value obtained so far can always survive. Before the global optimum is attained, however, it is a local optimum. If selective pressure for better-performing structures is high, structures similar to the best-performing structure will increase in number and eventually take over the population. This situation is premature convergence. Therefore, we need to reduce appropriately the selective pressure in the neighborhood of the best-performing structure to thin out structures similar to it. Eq. (1) can work to do such processing. The selection of structures on the basis of Eq. (1) is biased toward thinning out structures with smaller hamming distance from the best-performing structure and selecting structures with larger hamming distances from the best-performing structure. The larger bias produces greater diversity of structures in the population. This does not mean to neglect the selective pressure. As a result, there exists an appropriate selective pressure because the selection process is performed in order of the fitness values of the structures and better-performing structures can have an appropriate chance to be selected.

The degree of this bias is "externally" adjusted by the values of c and α. Their appropriate values need to be explored by trial and error according to the given problem. As demonstrated in the experiments described later, Eq. (1) is very suitable for controlling the diversity of the structures in the population so as to be in an appropriate condition by adjusting the values of c and α. DCGA is based on the idea that the selective pressure and population diversity should be externally controlled independently of the condition of the population, because the algorithm itself cannot recognize whether the population is in the region of a local optimum or in that of the global optimum.

In DCGA, the speed of convergence to the global optimum can be controlled indirectly by the user through the values of α and c of Eq. (1). This method may slow the convergence speed to the global optimum in some case, whereas it can be compensated and eventually improved by preventing the solution from getting stuck at a local optimum and stagnating.

The survival selection on the basis of the CPSS method introduces probabilistic fluctuations into the composition of structures in the population. I believe that such probabilistic fluctuations in the population are effective for escaping from a local optimum in a similar way to simulated annealing (SA). The selection process is performed in order of the fitness values of the structures, without considering the fitness value itself. This gives more chance of survival to current worse structures with fitness values below the average and of evolution into better structures. In the PES method [4], because the structures are deterministically selected in order of their fitness values (that is, $p_s = 1.0$ for all structures), the diversity of structures is often rapidly

lost, and this results in premature convergence. The CPSS method can avoid such a situation.

When a structure is represented by a bit string, binary coding or gray coding [19] is usually used. With DCGA, because the performance with gray coding is superior to that with binary coding, it is recommended that gray coding be used.

The methods employed in DCGA can work cooperatively to escape from a local optimum and to avoid premature convergence in the following way.

With the simple GA, better-performing structures can produce multiple offspring. Therefore, structures for a dominating local optimum can increase rapidly and eventually take over the population. On the other hand, with DCGA, each structure has only one chance to become a parent, irrespective of its performance. In addition, the same structures are eliminated and the number of structures similar to the best-performing one is restricted by selection according to the CPSS method. This can prevent structures in the population from becoming identical or nearly so, and eventually lead to avoid premature convergence.

In DCGA, structures that survived and the structure with the best fitness value obtained so far can always become parents and produce their offspring. Crossovers are always applied to diverse structures maintained in the population. When a pair of structures with a small distance are mated, their neighborhood can be examined to result in the local search. When a pair of structures with a large distance are mated, a region not yet explored can be examined to result in the global search. In such a way, local as well as global searches can be performed in parallel.

The structure with the best fitness value obtained so far always survives as a promising candidate to attain the global optimum, and its neighborhood can be examined by the local search. On the other hand, a current worse-performing structure can survive with a certain probability. This may give the structure a chance to produce an offspring with a fitness value close to the global optimum, if it is possible. This mechanism is similar to that of simulated annealing (SA) that can escape from a local optimum by accepting a solution based on a probability whose performance is worse than the present solution. In DCGA also, the solution can escape from a local optimum in a similar way to SA.

2.3 Features of DCGA

If we compare DCGA with Goldberg's method [2], the purpose and the method of realizing diversity of structures are essentially different. The purpose of the former is to attain only the global optimum efficiently, whereas that of the latter is to investigate many peaks of a multimodal function in parallel. The latter method follows the traditional GA, and the reproduction probability of a member of the population is adjusted by modifying the fitness value of the structure according to how many population members occupy a niche of the solution space.

3. Experiments on Performance in Function Optimization

The performance of DCGA for large-scale multimo-

Table 1. Summary of benchmark problems.

No.	Function equation	Bounded conditions Function name	Optimal value
1	$f_1 = \sum_{i=1}^{n} f_{si}$, $f_2 = \sum_{i=1}^{n} f_{si}$	Goldberg's tightly-ordered deceptive f. [4]	Maximum 300
2	f_{si} : f(000)=28 f(001)=26 f(010)=22 f(011)=0 f(100)=14 f(101)=0 f(110)=0 f(111)=30	Goldberg's loosely-ordered deceptive f. [4]	Maximum 300
3	$f_3 = 0.5 + \frac{0.5 - \sin^2\sqrt{x^2+y^2}}{[1+0.001(x^2+y^2)]^2}$	$-100 \le x, y \le 100$ Schaffer [20]	Maximum 1
4	$f_4 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$-30 \le x_i \le 30$ Ackley [23]	Minimum 0
5	$f_5 = \sum_{i=1}^{n}\left[-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right]$	$-512 \le x_i \le 511$ Schwefel [21]	Minimum $-418.982887 \cdot n$
6	$f_6 = 10 * n + \sum_{i=1}^{n}\left[x_i^2 - 10 * \cos(2\pi x_i)\right]$	$-5.12 \le x_i \le 5.11$ Rastrigin [21]	Minimum 0
7	$f_7 = 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\left[\cos\left(x_i / \sqrt{i}\right)\right]$	$-512 \le x_i \le 511$ Griewangk [21]	Minimum 0
8	$f_8 = \sum_{i=1}^{n-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + (1-x_i)^2\right] + 100\left(x_n^2 - x_1\right)^2 + (1-x_n)^2$	$-2.048 \le x_i \le 2.047$ Rosenbrock [21]	Minimum 0

Table 2. Definitions of symbols in subsequent Tables.

Symbol	Definition
N	Population size
n	Number of dimension
α	Exponent for probability function, Eq. (1)
c	Shape coefficient for probability function, Eq. (1)
CVR	Convergence rate (rate of successful runs)
AVFE	Average number of function evaluation times
SDFE	Standard deviation of function evaluation times
AVBF	Average value of best fitness values in all runs
MXFE	Maximum function evaluation times

dal functions that have been frequently used as benchmarks was tested and compared with those of existing leading methods. The functions used are summarized in Table 1. The dimension of the problem (n) and maximum function evaluation times (*MXFE*) were set according to the previous studies. With f_3, a bit string of length 22 for each dimension was used. With f_4, f_5, f_6, and f_7, a bit string of length 10 for each dimension was used. With f_8, a bit string of length 12 for each dimension was used. Gray coding was used except for f_1 and f_2. Bit-flip mutation was used. Two-point crossover for f_1, f_2 and f_3, and uniform crossover HUX [4] for the other functions were used. Combinations of best-performing parameter values including the size of population were examined by changing their values little by little. I performed 50 runs for f_1, f_2 and f_3, and 30 runs for the other functions per parameter set, changing seed values for the random number generator to initialize the population.

Table 2 shows the definitions of major symbols used in subsequent Tables. The performance was evaluated by the rate of instances out of the total runs in which the GA converged (*CVR*) and the average number of function evaluation times in those runs that converged (*AVFE*). The best results of the experiments are summarized in Table 3. With f_1, f_2, f_3, f_4, f_5, and f_6, the convergence rates were 1.0, whereas with f_7 and f_8, the convergence rates were not 1.0.

The performances of existing leading methods for each benchmark function were summarized in Table 4. In Yang's CGA, an error threshold = 10^{-3} was used as the convergence criterion. Also, Yang implemented evolutionary strategy (ES) and evolutionary programming (EP), and conducted experiments by the same conditions. For Bäck's ES and EP, results are averaged over 20 runs. In Mühlenbein's GA, the global minimum has been found to at least three digits of accuracy.

According to these results, the performances of GAs are superior to those of ESs and EPs for most of the test functions. In terms of DCGA and CHC, the former is superior to the latter for f_1, f_2, f_6 and f_8, and the latter is superior to the former for f_3, f_5, and f_7. DCGA is superior to Genitor for f_7. It appears that Yang's CGA and Mühlenbein's GA show good performance. However, the comparison with them is not fair, because their convergence criteria contain the error threshold.

4. Examination of Search Process

It is interesting to know how DCGA succeeds or fails in attaining the global optimum during the search process. In order to conjecture this, we need to know how DCGA works and of what structures the population is composed. Thus, in some cases where the algorithm succeeded or failed in attaining the global optimum, I examined the relationships between minimum (best), average, and maximum fitness values and generation number, and those between the ratios (h / L) of minimum, average, and maximum hamming distances to the length of the string and generation number.

Fig. 3 shows the case where DCGA succeeded in attaining the global optimum for griewangk's function (f_7). In both cases where DCGA succeeded or failed in attaining the global optimum, the best structure was trapped at a local minimum whose value is 0.0505. In the case of success, the algorithm could escape from the local optimum, whereas in the case of failure, the algorithm kept trapped there until *MXFE*. Comparing the case of success with the case of failure, we could not find the major difference between the ratios of hamming distances.

5. Discussion

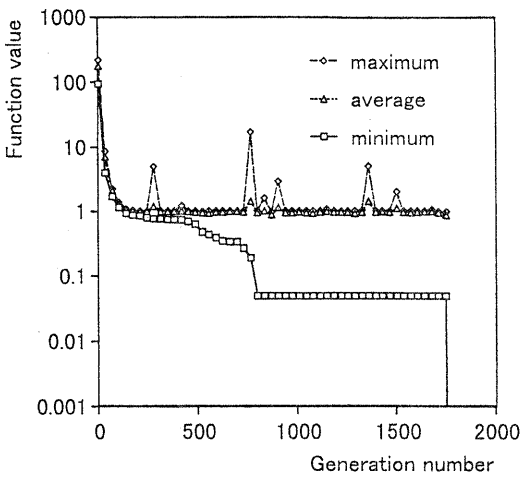
The examination of search process shows that during the search process, the population is composed of structures with considerably different hamming distances and thus the CPSS method effectively works. However, with griewangk's function (f_7) and Rosenbrock's function (f_8), the convergence rates were not 1.0. These cases show the limitation of performance of DCGA using ordinary crossover and bit-flip mutation operators. Comparing the case of success with the case of failure, we could not find the major difference in hamming distances between a structure and the structure with the best fitness and in fitness values of structures. In the case of failure, the best structure was trapped at a local minimum and kept trapped there until *MXFE*. These results show that in some cases, ordinary crossover and bit-flip mutation operators could not produce a structure that can escape from the local minimum. It appears that in harder problems, the combination of these operators does not have sufficient ability to explore regions that have not yet been examined. Hinterding [25] pointed out the limitation of bit-flip mutation as a reproduction operator and showed that Gussian mutation is superior to bit-flip mutation for most of the test functions [26]. Thus, em-

Table 3. Best results for each benchmark problem using DCGA.

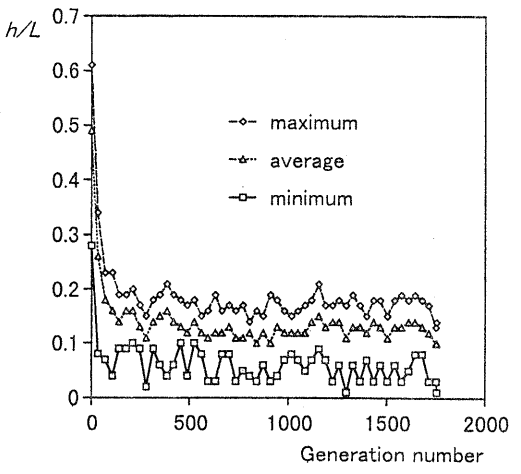
Func.	n	*MXFE*	N	α	c	p_m	*CVR*	*AVFE*	*SDFE*	*AVBF*
f_1	10	50000	4	0.51	0.33	0.008	1.0	6182	3452	300.0
f_2	10	50000	4	0.37	0.83	0.045	1.0	14996	6512	300.0
f_3	2	50000	12	0.5	0.234	0.022	1.0	16051	9000	1.0
f_4	30	100000	6	0.03	0.005	0.006	1.0	35477	7491	0.0
f_5	10	100000	2	0.0001	0.6	0.029	1.0	11428	5095	$4.18982887 \cdot 10^3$
	20	200000	2	0.0001	0.6	0.018	1.0	62708	18424	$8.37965774 \cdot 10^3$
f_6	20	300000	2	0.11	0.003	0.006	1.0	140207	37872	0.0
f_7	10	500000	46	0.21	0.01	0.006	0.87	160298	122713	$7.140 \cdot 10^{-3}$
	20	500000	50	0.21	0.01	0.0021	0.77	264599	106174	$1.145 \cdot 10^{-2}$
f_8	6	200000	28	0.2	0.008	0.012	0.53	77723	24167	2.77
	8	500000	34	0.204	0.0005	0.01	0.5	238829	116860	3.96
	10	5000000	42	0.2	0.002	0.011	0.47	2286790	812327	5.28

Table 4. Best results for each problem using existing algorithms.

Func.	n	Reference, Method	Run No.	*MXFE*	N	*CVR*	*AVFE*	*AVBF*
f_1, f_2	10	CHC[4], GA	50	50000	50	1.0	20960	300.0
f_3	2	CHC[4], GA	50	50000	50	1.0	6496	—
f_4	30	Yang[10], GA	20	200000	50	1.0	43068	—
		Bäck[23], ES	20	100000	30	—	—	$1.62 \cdot 10^{-3}$
		Yang[10], ES	20	200000	—	0.85	35514	—
		Bäck[23], EP	20	100000	200	—	—	1.98
f_5	10	Mühlenbein[22], GA	50	—	8·20	0.92	8699	—
		CHC[24], GA	30	500000	50	1.0	9803	—
		Yang[10], GA	20	200000	50	1.0	71273	—
		Yang[10], ES	20	200000	—	0.04	7875	—
		Yang[10], EP	20	200000	—	0.0	—	—
	20	CHC[24], GA	30	500000	50	1.0	17123	—
f_6	20	Mühlenbein[22], GA	50	—	8·20	1.0	9900	—
		CHC[24], GA	30	500000	50	1.0	158839	0.0
		Yang[10], GA	20	200000	50	1.0	81440	—
		Yang[10], ES	20	200000	—	0.0	—	—
		Yang[10], EP	20	200000	—	0.26	342100	—
f_7	10	Mühlenbein[22], GA	50	—	16·40	1.0	59520	—
		CHC[24], GA	30	500000	50	1.0	51015	0.0
		Genitor[12], GA	30	500000	1000	0.83	92239	$5.96 \cdot 10^{-3}$
		Yang[10], GA	20	200000	100	1.0	126785	—
		Yang[10], ES	20	200000	—	0.02	23835	—
		Yang[10], EP	20	200000	—	0.0	—	—
	20	CHC[24], GA	30	500000	50	1.0	50509	0.0
		Genitor [12], GA	30	500000	1000	0.57	104975	$2.40 \cdot 10^{-2}$
f_8	6	CHC[21], GA	30	5000000	50	0.0	—	5.939
	8	CHC[21], GA	30	5000000	50	0.067	—	7.391
	10	CHC[21], GA	30	5000000	50	0.033	—	9.569
		Yang[10], GA	20	450000	30	1.0	356991	—
		Yang[10], ES	20	400000	—	0.65	274726	—
		Yang[10], EP	20	400000	—	0.0	—	—



(a)



(b)

Fig. 3. Conditions of the population in the case where DCGA succeeded in attaining the global optimum for Griewangk's function (n = 10). (a) Generation number vs. function value, (b) Generation number vs. ratio of hamming distance (h / L) between a structure and the structure with the best fitness value.

ploying more powerful mutation operators may be a promising way to further improve the performance of DCGA.

According to the above results, the combinations of best-performing parameter values are considerably different for the given problems. In this sense, DCGA is sensitive to the parameter values and they must be tuned by trial and error for each given problem.

The best results using DCGA obviously shows that there exists an optimum diversity of the structures in the population according to the given problem. The value of p_{s0}, which is p_s for h = 0, represents the magnitude of the selection probability, and a smaller p_{s0} can produce a higher diversity of structures in the population. The values of p_{s0} in the best results are 0.57 for tightly ordered deceptive function, 0.93 for loosely ordered deceptive function, 0.48 for Schaffer's function, 0.85 for Ackley's function, 1.0 for Schwefel's function, 0.53 for Rastrigin's function, 0.38 for Griewangk's function, 0.29 for Rosenbrock's function (n = 10), respectively. It appears that a harder problem requires a larger diversity of structures in the population.

The best results using DCGA obviously show that there exists an optimum population size. It seems that a harder problem requires a larger population size. As a whole, the optimum population size is small and good performance is obtained with such a small population. This indicates that if structures that keep appropriate distances from the current best structure are distributed in the solution space, only a small number of such structures are sufficient to attain the global optimum. Such a characteristic of DCGA has many advantages in the stage of implementing and using it in practical applications.

It has been recognizes that real-coded GAs are more effective than binary-coded GAs [9, 27] in some problems. The CPSS method can be applied to the real-coded GAs by using some distance measure such as Euclidean distance instead of hamming distance. At present, I am conducting such an extension of DCGA.

6. Conclusions

Within the range of the above experiments using conventional crossover and bit-flip mutation operators, the following conclusions can be drawn.

① DCGA is effective and efficient for large-scale multimodal functions. The optimum population size is small and good performance is obtained with such a small population.

② There is some room to further improve the performance, because with some hard problems, the convergence rates were not 1.0.

③ In some problems, the performances of DCGA were superior to those of existing leading methods. According to these results, DCGA may be a promising competitor to existing leading methods.

References

[1] M. L. Mauldin, "Maintaining Diversity in Genetic Search," in *Proc. of the National Conference on Artificial Intelligence*, 1984, pp. 247-250.

[2] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization," in *Proc. of the Second International*

Conference on Genetic Algorithms, Lawrence Erlbaum, 1987, pp. 41-49.

[3] S. W. Mahfoud, "Crowding and Preselection Revisited," in *Parallel Problem Solving from Nature 2*, R. Männer Ed., Amsterdam: North-Holland, 1992, pp. 27-36.

[4] L. J. Eshelman, "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," in *Foundation of Genetic Algorithms*, G. J. E. Rawlins Ed., California: Morgan Kaufmann, 1991, pp. 265-283.

[5] L. J. Eshelman and J. D. Schaffer, "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest," in *Proc. of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991, pp. 115-122.

[6] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24, No. 4, pp. 656-667, April, 1994.

[7] H. Shimodaira, "Proposal of a Genetic Algorithm Using Large Mutation Rates and Population-Elitist Selection (GALME)," 情報処理学会研究報告, 人工知能, 106-14, 1996, pp. 99-106.

[8] H. Shimodaira, "A New Genetic Algorithm Using Large Mutation Rates and Population-Elitist Selection (GALME)," in *Proc. of the International Conference on Tools with Artificial Intelligence*, IEEE Computer Society, 1996, pp. 25-32.

[9] M. Sefrioui and J. Périaux, "Fast Convergence Thanks to Diversity," in *Evolutionary Programming V, Proc. of the Fifth Annual Conference on Evolutionary Programming*, 1996, pp. 313-321.

[10] J. Yang, J. Horng and C. Kao, "A Continuos Genetic Algorithms for Global Optimization," in *Proc. of the Seventh International Conference on Genetic Algorithms*, Morgan Kaufmann, 1997, pp. 230-237.

[11] J. R. Greene, "A Role for Simple, Robust 'Black-box' Optimizers in the Evolution of Engineering Systems and Artifacts," in *Proc. of the Second IEE/IEEE Int. Conference on Genetic Algorithms in Engineering Systems*, 1997, pp. 427-432.

[12] D. Whitley, R. Beveridge, C. Graves and K. Mathias, "Test Driving Three 1995 Genetic Algorithms: New Test Functions and Geometric Matching," *Journal of Heuristics*, Vol. 1, pp. 77-104, 1995.

[13] H. Shimodaira, "DCGA: A Diversity Control Oriented Genetic Algorithm," in *Proc. of the Second IEE/IEEE Int. Conference on Genetic Algorithms in Engineering Systems*, 1997, pp. 444-449.

[14] H. Shimodaira, "DCGA: A Diversity Control Oriented Genetic Algorithm," in *Proc. of the Ninth IEEE Int. Conference on Tools with Artificial Intelligence*, 1997, pp. 367-374.

[15] H. Shimodaira, "A Diversity-Control-Oriented Genetic Algorithm (DCGA): Development and Initial Results," 情報処理学会論文誌, Vol. 40, No. 6, 1999.

[16] H. Shimodaira, "A Diversity-Control-Oriented Genetic Algorithm (DCGA): Development and Experimental Results," in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-99) Volume 1*, Morgan Kaufmann, 1999, pp.603-611.

[17] http://www.hi-ho.ne.jp/shimo-hi/new_ga.htm

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Massachusetts: MIT Press, 1992, p. 111.

[19] R. A. Caruana and J. D. Schaffer, "Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms," in *Proc. of the 5th Int. Conference on Machine Learning*, Morgan Kaufmann, 1988, pp. 153-161.

[20] J. D. Schaffer, R. A. Caruana, L. J. Eshelman and R. Das, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization," in *Proc. of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989, pp. 51-60.

[21] D. Whitley, K. Mathias, S. Rana and J. Dzubera, "Building Better Test Functions," in *Proc. of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1995, pp. 239-246.

[22] H. Mühlenbein, M. Schomisch and J. Born, "The Parallel Genetic Algorithm as Functional Optimizer," *Parallel Computing*, Vol. 17, pp. 619-632, 1991.

[23] T. Bäck and H. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, Vol. 1, No. 1, pp. 1-23, 1993.

[24] K. E. Mathias and D. Whitley, "Transforming the Search Space with Gray Coding," in *Proc. of the first IEEE Conference on Evolutionary Computation*, 1994, pp. 513-518.

[25] R. Hinterding, H. Gielewski and T. C. Peachery, "The Nature of Mutation in Genetic Algorithms," in *Pro. of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1995, pp. 65-72.

[26] R. Hinterding, "Gussian Mutation and Self-adaptation for Numeric Genetic Algorithms," in *Pro. of the 1995 IEEE International Conference on Evolutionary Computation*, 1995, pp. 384-389.

[27] L. J. Eshelman and J. D. Schaffer, "Real-Coded Genetic Algorithms and Interval-Schemata," in *Foundations of Genetic Algorithms 2*, L. D. Whitley Ed., California: Morgan Kaufmann, 1993, pp. 187-202.