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

Hisashi Shimodaira
Faculty of Information and Communications, Bunkyo University
2-2-16 Katsuradai, Aoba-Ku, Yokohama-City, Kanagawa 227-0034, Japan
E-mail: shimo-hi@hi-ho.ne.jp

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 particular selection probability to maintain the diversity of the structures. The major feature is that the distance between a structure and the best performance structure is used as the primary selection criterion and it is applied on the basis of a probabilistic function that produces a larger selection probability for a structure with a larger distance. 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 categories: 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, Whitley's GENITOR [3] that employs one-at-a-time reproduction and rank-based allocation of reproduction trials, Eshelman's CHC [4] that employs population-elitist selection method with the incest avoiding and the restart functions, Mahfoud's deterministic crowding method [5] in which the selection is performed between two parents and their offspring on the basis of the phenotypic similarity measure, Mori's thermodynamical selection rule [6] which adopts the concepts of temperature and entropy as in the simulated annealing, and Mengshoel's probabilistic crowding (integrated tournament algorithms) [7].

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 results reported in these studies are not necessarily satisfactory. Therefore, I have developed a new genetic algorithm called DCGA (Diversity-Control-oriented Genetic Algorithm) that belongs to the former category [8, 9, 10, 11, 12].

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 particular selection probability. The major feature of DCGA is that the distance between a structure and the best performance structure is used as the primary selection criterion and it is applied on the basis of a probabilistic function that produces a larger selection probability for a structure with a larger 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 described in the previous papers [8, 9, 10, 11, 12], 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 struc-

0-7803-6657-3/01/$10.00 ©2001 IEEE

```
begin;
  t=0;
  initialize population P(t);
  evaluate structures in P(t);
  while (termination condition not satisfied) do;
  begin;
    t=t+1;
    select 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 produce two offspring to 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 N structures including the structure with the
        best fitness value from M(t) to form the next pop-
        ulation 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
            in order of their fitness values;
        (3) if the number of selected structures is smaller
            than N, introduce new structures by the
            difference of the numbers;
  end;
end;
```

Fig. 1 The skeleton of DCGA

tures (chromosomes) 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, 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), two offspring are produced and C(t) is formed. 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, 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, are as follows:

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:

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

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 whose value is in the range of [0.0, 1.0], and α is the exponent. In the selection process, a uniform random number in the range of [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 randomly generated as in the initial population are introduced by the difference of the numbers.

If the structure is represented as a bit string, the hamming distance between two structures can be calculated by the usual way. With a combinatorial optimization problem such as the traveling salesman problem, also, the extended hamming distance can be calculated as a minimum value of numbers of pairs that have different cities, when the cities of the two tours are paired each other in order of their path representations [9, 10, 11]. Thus, DCGA is applicable to all sorts of optimization problems by employing the definition of a proper distance measure between two structures.

2.2 Empirical and Theoretical Justification

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

In DCGA, the structure with best performance obtained so far always survives intact into the next generation. Therefore, large mutation rates can be used and crossover can be always 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, and low mutation rates are generally used. Also, crossover is applied with a certain rate smaller than 1.0.

Duplicate structures reduce the diversity of the structures in the population and often cause premature convergence [3], because the same structures can produce a large number of offspring with the same structure in the next generation and this result in premature domination of the population by the best-performing structure. Therefore, it is effective to eliminate duplicate structures in order to avoid premature convergence. The effectiveness of avoiding duplicate structures was shown by some experiments in the previous papers [9, 10, 11].

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

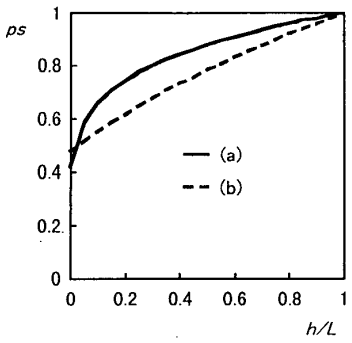


Fig. 2. Example curves of Eq. (1). (a) $\alpha = 0.19$, $c = 0.01$; (b) $\alpha = 0.5, c = 0.234$.

recognize whether the population is in the region of a local optimum or in that of the global optimum.

As long as the structure with the best fitness value does not reach the global optimum, it is a local optimum. If the selective pressure for better-performing structures is too high, structures similar to the best-performing structure will increase in number and eventually take over the population. This situation is premature convergence, if it is a local optimum. 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. Example curves of Eq. (1) are shown in Fig. 2. 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. As a result, the population is composed of various structures as demonstrated in Section 4.

The larger bias produces the greater diversity of structures in the population. The degree of this bias is "externally" adjusted by the values of c and α in Eq. (1). 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.

In the selection for survival, the fitness values of the structures themselves are not considered. However, this does not mean to neglect the 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, as a result, there exists an appropriate selective pressure determined by the value of the selection probability.

We can produce an appropriate selective pressure according to the given problem. That is, for a simple function with few local optima, higher selective pressure can be produced with a larger value of c and / or a smaller value of α For a complicated function with many local optima, lower selective pressure can be produced with a smaller value of c and / or a larger value of α. The selection with $c = 1.0$ in DCGA is the same as $(N + N)$-selection in the evolution strategies and the population-elitist selection in CHC [4].

In DCGA, the speed of convergence to an optimum (although it may not be the global optimum) can be controlled indirectly by the user through the values of α and c in 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.

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.

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. 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 more chance of evolution into better structures. In the population-elitist selection method, 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. The superiority of the CPSS method to the population-elitist selection method was shown by some experiments in the previous papers [8, 9, 10, 11].

The introduction of new structures occurs during iteration when the diversity of structures in the population happens to become smaller. This is equivalent to the introduction of very large mutations into the population, and can work effectively to restore the diversity automatically.

When a structure is represented by a bit string, binary coding or gray coding [13] is usually used. With DCGA, because the performance with gray coding is superior to that with binary coding [8, 9, 10, 11], it is recommended that gray coding be used.

The methods employed in DCGA can work cooperatively for the solution 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 the selection with 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

The major feature of DCGA is that the distance between a structure and the best performance structure is used as the primary selection criterion and it is applied on the basis of a probabilistic function that produces a larger selection probability for a structure with a larger distance.

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 with as a smaller population as possible, whereas that of the latter is to investigate many peaks of a multimodal function in parallel. The time complexities of the distance calculation are as follows. With DCGA, it is O(N) per generation. With CHC [4], it is also O(N) per generation. With Mauldin's method [1], it is O(N) per structure. With Goldberg's method [2], it is O(N^2) per generation. Therefore, the computational cost of maintaining the diversity of structures is much lower for DCGA than for Mauldin's and Goldberg's methods.

A shortcoming of DCGA is that the number of parameters to be tuned is three (mutation rate, and α and c in Eq. (1)) and they must be tuned trial and error according to the given problem.

Table 1. Summary of benchmark functions.

No.	Function name	Bits / dim.	Ref.
F_1	De Jong f. F_1	10	14
F_2	De Jong f. F_2	12	14
F_3	De Jong f. F_3	10	14
F_4	De Jong f. F_4	8	14
F_5	De Jong f. F_5	17	14
F_6	Goldberg's order-3 tightly-ordered deceptive f.	3	4
F_7	Goldberg's order-3 loosely-ordered deceptive f.	3	4
F_8	Sine envelope sine wave f.	22	16
F_9	Generalized Ackley f.	10	17
F_{10}	Schwefel f.	10	14
F_{11}	Rastrigin f.	10	14
F_{12}	Griewank f.	10	14
F_{13}	Expanded Rosenbrock f.	12	Eq.(2)

3. Experiments on Performance in Function Optimization

The performances of DCGA for various functions that have been frequently used as benchmarks were tested and compared with those of existing leading methods. The functions used are summarized in Table 1. F_{13} is the expanded Rosenbrock function [15] as follows:

$$F_{13} = \sum_{i=1}^{n-1} \left[100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \right] + 100(x_n^2 - x_1)^2 + (1 - x_n)^2 \qquad (2)$$

All the problems except for F_6 and F_7 are minimum-seeking problems. The binary string representing a structure was transformed to real numbers in the phenotype so that for each coordinate, the corresponding real number matches exactly the coordinate value which gives the global optimum of the function. Exact global optimums in these functions except for F_4 were explored with the optimality threshold considering only round-off errors. In F_4, the optimality threshold $\varepsilon = 10^{-3}$ was used. The dimension of the problem (n) and the maximum number of function evaluation (*MXFE*) were set according to the previous studies [14, 15, 18]. Gray coding was used except for F_6 and F_7. Bit-flip mutation was used. Two-point crossover for F_6, F_7 and F_8, and uniform crossover HUX [4] for the other functions were used. I performed 50 runs for F_6, F_7 and F_8, and 30 runs for the other functions per parameter set, changing seed values for the random number generator to initialize the population. The run was continued until the global optimum was attained by at least one structure (I call this the success) or until *MXFE* was reached.

The combination of best-performing parameter values including the population size was examined by changing their values little by little as follows. First, we perform runs by changing roughly the value of a parameter and fixing the values of the other parameters that are set initially by conjecture and find out an approximately optimal value of the parameter by observing the best fitness value.

Table 2. Definitions of symbols in the 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)
SCR	Success rate (rate of successful runs)
AVFE	Average value of function evaluation numbers in the successful runs
SDFE	Standard deviation of function evaluation numbers in the successful runs
AVBF	Average value of best fitness values in all runs
MXFE	Maximum number of function evaluation
NR	Total number of runs
ε	Terminating condition (Optimality threshold)

Table 3. Best results (exact solution except for F_4; for F_4, $\varepsilon = 10^{-3}$) for each benchmark function using DCGA.

F.	n	*MXFE*	N	α	c	p_m	*SCR*	*AVFE*	*SDFE*	*AVBF*
F_1	3	674	2	any	1.0	0.035	1.0	443	123	0.0
F_2	2	20741	6	0.2	0.012	0.04	1.0	8106	5471	0.0
F_3	5	3776	16	0.229	0.46	0.035	1.0	1509	738	-30.0
F_4	30	15840	16	0.095	0.0014	0.006	1.0	7896	3734	$4.185 \cdot 10^{-4}$
F_5	2	1212	12	0.1	0.002	0.058	1.0	555	221	0.998003838
F_6	10	18880	4	0.51	0.33	0.008	1.0	6182	3452	300.0
F_7	10	34889	4	0.37	0.83	0.045	1.0	14996	6512	300.0
F_8	2	46301	12	0.5	0.234	0.022	1.0	16051	9000	0.0
F_9	30	54009	6	0.03	0.005	0.006	1.0	35477	7491	0.0
F_{10}	10	25676	2	0.0001	0.6	0.029	1.0	11428	5095	$-4.18982887 \cdot 10^3$
	20	102306	2	0.0001	0.6	0.018	1.0	62708	18424	$-8.37965774 \cdot 10^3$
F_{11}	20	249182	2	0.11	0.0008	0.006	1.0	139102	44484	0.0
F_{12}	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_{13}	6	500000	28	0.2	0.008	0.012	0.53	77723	24167	2.77
	8	1000000	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. Results for the Akley function (F_9) using various combinations of parameter values (*MXFE* = 80000).

N	α	c	p_m	*SCR*	*AVFE*	*SDFE*
6	0.03	0.005	0.006	1.0	35477	7491
4	0.03	0.005	0.006	1.0	36779	8697
8				1.0	44480	12596
6	0.02	0.005	0.006	1.0	39825	11923
	0.04			1.0	41471	10380
6	0.03	0.0005	0.006	1.0	36810	10946
		0.05		1.0	43161	10202
6	0.03	0.005	0.004	1.0	39465	10828
			0.008	1.0	52137	13256

We can obtain approximately optimal values of all the parameters by repeating this processing. Next, we perform the same processing near the approximately optimal values of the parameters by changing their values little by little and can obtain the optimal value of each parameter. By such processing, we can obtain the optimum value of the function as well as the combination of best-performing parameter values fairly easily without testing unpromising combinations of parameter values.

The performance was evaluated by the rate of successful runs out of the total runs (*SCR*) and the average value of function evaluation numbers in the successful runs (*AVFE*). Table 2 shows the definitions of major symbols used in the subsequent Tables. The best results of the experiments are summarized in Table 3. For *MXFE*, if the success rate is 1.0 with the prescribed *MXFE*, the actual maximum value of function evaluation numbers in all runs is indicated. With F_1 — F_{11}, the success rate was 1.0, whereas with F_{12} and F_{13}, the success rate was not 1.0. In order to demonstrate the sensitivity of the performance to the parameter settings, Table 4 shows the results of runs for the generalized Ackley function (F_9) using a different combination of parameter values changing each value a little near the best ones.

The performances of existing methods for each benchmark function were summarized in Table 5. According to these results, with functions F_1, F_2, F_4, F_5, F_6, F_7, F_{11}, F_{13}, the performances of DCGA are better than those of CHC that is one of the best GAs. Also, for F_{12}, DCGA is superior to Genitor that is also one of the best GAs.

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 realize this, we need to know how DCGA works and of what structures the population is composed. Thus, in some cases where DCGA succeeded or failed in attaining the global optimum, I examined the relationships between minimum (best), average, and maximum fitness values and generation number, and relationships between minimum, average, and maximum values of the ratio h / L and generation number.

Fig. 3 and Fig. 4, respectively, show the case where DCGA succeeded or failed in attaining the global optimum for the Griewank function (F_{12}, $n = 10$) under the condition shown in Table 3. In the case of success, the best structure was trapped at a local minimum whose function value is 0.0498, whereas it could escape from the local optimum and reach the global optimum. In the case of failure, the best structure was trapped at a local minimum whose function value is 0.0488 and it kept trapped there until *MXFE*. In both cases, the diversity of structures in both genotype and fitness value were maintained.

5. Discussion

The examination of search process shows that during the search process, the population is composed of structures with considerably

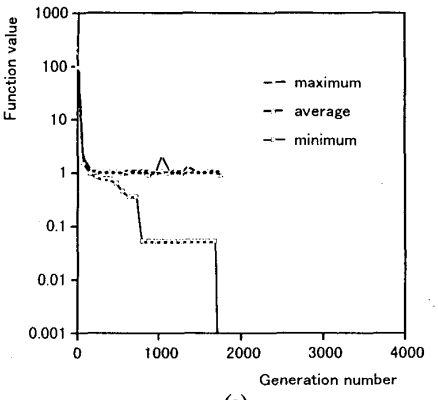
Table 5. Best results for each function using existing algorithms.

F.	n	Reference, Method	*NR*	*MXFE*	*N*	*SCR*	*AVFE*	*AVBF*
F_1	3	SGA [4], Simple GA	50	50000	—	1.0	805	—
		CHC [18], GA	30	200000	50	1.0	1126	—
F_2	2	SGA [4], Simple GA	50	50000	—	1.0	9201	—
		CHC [18], GA	30	100000	50	1.0	9455	—
F_3	5	SGA [4], Simple GA	50	50000	—	1.0	1270	—
		CHC [18], GA	30	—	50	1.0	1265	—
F_4	30	SGA [4], Simple GA	50	50000	—	1.0	2228	—
		CHC [18], GA	30	100000	50	1.0	16335	—
F_5	2	SGA [4], Simple GA	50	50000	—	1.0	1719	—
		CHC [18], GA	30	50000	50	1.0	733	—
F_6	10	SGA [10, 11], Simple GA	50	50000	80	0.1	34720	294.4
		CHC [4], GA	50	50000	50	1.0	20960	—
F_7	10	SGA [10, 11], Simple GA	50	100000	110	0.4	74591	297.9
		CHC [4], GA	50	50000	50	1.0	20960	—
F_8	2	SGA [10, 11], Simple GA	50	50000	48	0.28	14983	$1.29 \cdot 10^{-4}$
		CHC [4], GA	50	50000	50	1.0	6496	—
F_{10}	10	CHC [18], GA	30	500000	50	1.0	9803	—
	20	CHC [18], GA	30	500000	50	1.0	17123	—
F_{11}	20	CHC [18], GA	30	500000	50	1.0	158839	—
F_{12}	10	CHC [18], GA	30	500000	50	1.0	51015	—
		Genitor [14], GA	30	500000	1000	0.83	92239	$5.96 \cdot 10^{-3}$
	20	CHC [18], GA	30	500000	50	1.0	50509	—
		Genitor [14], GA	30	500000	1000	0.57	104975	$2.40 \cdot 10^{-2}$
F_{13}	6	SGA [15], Simple GA	30	5000000	—	0.1	—	5.345
		CHC [15], GA	30	5000000	50	0.0	—	5.939
	8	SGA [15], Simple GA	30	5000000	—	0.23	—	6.071
		CHC [15], GA	30	5000000	50	0.067	—	7.391
	10	SGA [15], Simple GA	30	5000000	—	0.2	—	7.919
		CHC [15], GA	30	5000000	50	0.033	—	9.569

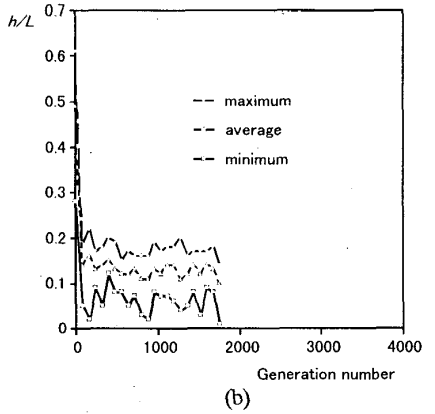
different hamming distances and thus the CPSS method effectively works. However, with the Griewank function (F_{12}) and the expanded Rosenbrock function (F_{13}), the success rate was not 1.0. In the case of failing in attaining the global optimum, the best structure was trapped at a local minimum and kept trapped there until *MXFE*, because crossover and bit-flip mutation operators could not produce a structure that can escape from the local minimum. These results show that in harder problems, the combination of these operators does not have sufficient ability to explore regions that have not yet been examined. Hinterding [19] pointed out the limitation of bit-flip mutation as a reproduction operator and showed that Gaussian mutation is superior to bit-flip mutation for most of the test functions [20]. Thus, employing more powerful mutation operators such as Gaussian mutation or Cauchy mutation [21] may be a promising way to improve further the performance of DCGA. Also, annealing-based mutation [22] and restart function [4] may be effective for the solution to escape from local minimums.

The best results show obviously 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 as follows: 1.0, 0.41, 0.84, 0.54, and 0.54 for the De Jong functions F_1— F_5, respectively; 0.57 for the tightly ordered deceptive function F_6; 0.93 for the loosely ordered deceptive function F_7; 0.48 for the Schaffer function F_8; 0.85 for the Ackley function F_9; 1.0 for the Schwefel function F_{10}; 0.53 for the Rastrigin function F_{11}; 0.38 for the Griewank function F_{12}; 0.29 for the expanded Rosenbrock function F_{13} (n = 10); respectively. It appears that a harder problem requires a larger diversity of structures in the population.

The best results show obviously 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. For

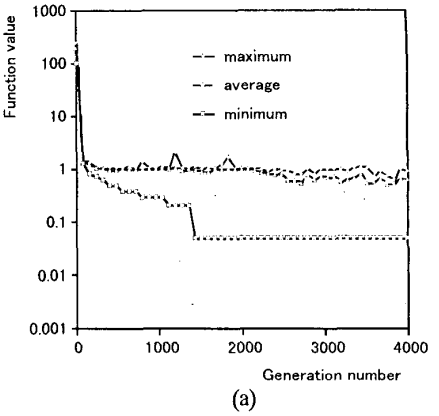


(a)

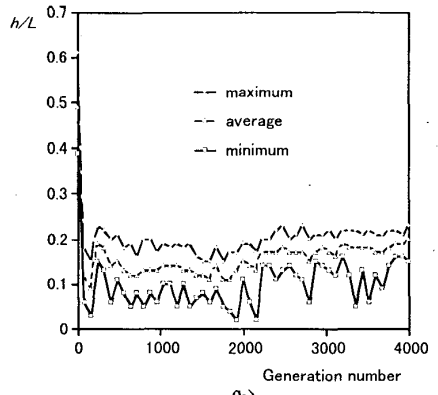


(b)

Fig. 3. Conditions of the population in the case where DCGA succeeded in attaining the global optimum for the Griewank function (F_{12}). (a) Function value vs. generation number, (b) Ratio h/L vs. generation number.



(a)



(b)

Fig. 4. Conditions of the population in the case where DCGA failed in attaining the global optimum for the Griewank function (F_{12}). (a) Function value vs. generation number, (b) Ratio h/L vs. generation number.

example, we can implement it using a small memory capacity.

Because according to the above results, the combinations of best-performing parameter values including the population size are considerably different for the given problems, we need to search for an optimum parameter set for each new problem to achieve the best performance. This is not a peculiar problem to DCGA. That no single robust parameter set exists that is suitable for a wide rang of functions has been demonstrated for the GA theoretically [23] and for the Evolutionary Programming experimentally [24]. With DCGA, the best parameter set can be obtained fairly easily in the process of exploring the optimum value of the function by employing the method described in Section 3.

6. Conclusions

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

① For the standard benchmark problems except the Griewank and expanded Rosenbrock functions, the success rate was 1.0. DCGA is promising as a function optimization tool, though there is some room to improve further the performance in some harder problems.

② The optimum population size is small and good performance is obtained with such a small population.

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

In order to improve further the performance, I am now exploring the possibilities for employing restart functions and mutation operators other than bit-flip mutation. The results will be reported elsewhere in the near future. In future, the evaluation of the performance in optimization of large-scale multimodal functions and the theoretical analysis of the search process is required.

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] D. Whitley, "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproduction Trials is Best", in *Proc. of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989, pp. 116-121.

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

[6] N. Mori and J. Yoshida, "A Thermodynamical Selection Rule for the Genetic Algorithm", in *Pro. of the second IEEE International Conference on Evolutionary Computation*, pp. 188-192, 1995.

[7] O. J. Mengshoel and D. E. Goldberg, "Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement", in *Pro. of the Genetic and Evolutionary Computation Conference (GECCO-99)*, Vol.1, pp.409-416, 1999.

[8] H. Shimodaira, "DCGA: A Diversity Control Oriented Genetic Algorithm," in *Proc. of the Second IEE/IEEE Int. Conference on Genetic Algorithms in Engineering Systems (GALESIA'97)*, 1997, pp. 444-449.

[9] 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.

[10] H. Shimodaira, "A Diversity-Control-Oriented Genetic Algorithm (DCGA): Development and Initial Results," Transactions of Information Processing Society of Japan, Vol. 40, No. 6, 1999.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

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

[21] X. Yao and Y. Liu, "Fast Evolution Strategies", in *Evolutionary Programming VI*, Springer, 1997, pp. 151-161.

[22] 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.

[23] W. E. Hart and R. K. Belew, "Optimizing an Arbitrary Function is Hard for the Genetic Algorithm", in *Proc. of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991, pp. 190-195.

[24] K. E. Mathias, J. D. Schaffer and L. J. Eshelman, "The Effects of Control Parameters and Restarts on Search Stagnation in Evolutionary Programming", *Parallel Problem Solving from Nature - PPSN V*, Springer, 1998, pp. 398-407.