Vol. 40 No. 6 Transactions of Information Processing Society of Japan June 1999

Regular Paper

A Diversity-Control-oriented Genetic Algorithm (DCGA): Development and Initial Experimental Results

HISASHI SHIMODAIRA†

In genetic algorithms, in order to attain the global optimum without getting stuck at a local optimum, appropriate diversity of the structures in the population needs to be maintained. I propose a new genetic algorithm called DCGA (Diversity-Control-oriented Genetic Algorithm) to attain this goal. In the 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, and is larger for structures with larger hamming distances. 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 my experiments, the DCGA showed a markedly superior performance to the simple GA and it seems to be a promising competitor of previously proposed algorithms.

1. Introduction

Genetic algorithms (GAs) are a promising means for function optimization. Methods for function optimization are required to attain the global optimum without getting stuck at a local optimum. For multimodal functions, because the performance of the simple GA is poor in this respect, various studies have been carried out to improve the performance of the GA. The major ones are as follows.

For the simple GA, Baker[1] observed that premature convergence (convergence to a local optimum) often occurs after an individual or a small group of individuals contributes a large number of offspring to the next generation. Booker[2] mentioned that a large number of offspring for one individual means fewer offspring for the rest of the population, and when too many individuals have no offspring at all, the result is a rapid loss of diversity and premature convergence. What is needed to handle premature convergence is to prevent this situation.

In the paradigm of the traditional GA, Goldberg, et al.[3] proposed a method of sharing function that mitigates this issue by allowing the formation of species in niches. This mechanism modifies the reproduction probability of a population member by adjusting the fitness value of the structures according to how many population members occupy a niche of the solution space. Although the method has proved effective, it is computationally very expensive, because the distance between each two structures in the population needs to be calculated.

Also in the paradigm of the traditional GA, Srinivas, et al.[4] proposed the use of adaptive probabilities of crossover and mutation to realize the twin goals of maintaining diversity in the population and sustaining the convergence capacity of the GA. In his algorithm, the probabilities of crossover and mutation are varied depending on the fitness values of structures. High fitness structures are protected, while structures with subaverage fitness value are totally disrupted. However, because the selection for reproduction is biased toward selecting the better-performing individuals, premature convergence often occurs and is essentially inevitable as indicated in the results of Srinivas.

Eshelman[5] proposed an algorithm employing a highly destructive uniform crossover (HUX) and the Population-Elitist Selection (PES) method that is based on cross-generational deterministic rank-based survival selection. In the reproduction stage, two candidate structures are selected for mating. In order to maintain diversity, the hamming distance between them is calculated, and if half that distance does not exceed a difference threshold, they are not mated and deleted from the population. The difference threshold is automatically reduced as the population converges. Many studies have shown that the performance

† Department of Information and Communication, Bunkyo University

of the CHC in the standard benchmark tests is extremely good and robust [6),7)], whereas the algorithm is not so easy to use, because it contains the restart function.

In order to attain the global optimum without getting stuck at a local optimum, it is essential to control the diversity of the structures in the population during the search so that local searching and global searching are performed in a balanced way. To attain these twin goals, I have developed a new genetic algorithm called DCGA (Diversity-Control-oriented Genetic Algorithm) [8),9)]. In the DCGA, the structures for 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 the candidate structure and the structure with the best fitness value. The selection probability is larger for structures with larger hamming distances. 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 my experiments, the DCGA outperformed the simple GA and seems to be a promising competitor of the previously proposed algorithms.

This paper describes the DCGA and presents the results of experiments to show the effectiveness of the proposed method. The results are compared with those for the simple GA and for the promising previous studies.

2. The Simple GA

The outline of the simple GA is described to facilitate the later explanation. In the simple GA, the following processes are performed:

(1) The number N of individuals in the population is constant, and the population is initialized by using random numbers.

(2) In the reproduction stage, structures are selected from the present population $\mathrm{P}(t-1)$ and recombined to form the offspring population $\mathrm{C}(t)$, where t is the generation and a structure is the genotype of an individual. The selection for reproduction (select_r) is biased toward selecting the better-performing individuals. The recombination is performed by using crossover on the basis of probability. A low rate of mutation is used in the recombination stage to maintain population diversity.

(3) The selection for survival (select_s) is usually unbiased, typically replacing the entire parent population $\mathrm{P}(t-1)$ with the child population $\mathrm{C}(t)$.

3. DCGA

In order to improve the performance of GAs, the algorithm needs to have the ability to robustly explore the solution space to find out the best region containing the global optimum (global search) and to escape from a local optimum when it gets stuck there. Attaching importance to only current better-performing structures may result in premature convergence. On the other hand, a current worse-performing structure may have a greater potential for evolving toward a better future structure to attain the global optimum. The idea motivating my research is to exploit these worse-performing structures, instead of discarding them, by maintaining the diversity of structures in the population. In addition, it needs to exploit the best structure obtained so far, because it may be in the region containing the global optimum. The DCGA was devised to achieve these twin goals.

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

The details of the selection for survival, select_s, are as follows:

(1) Duplicate structures in $\mathrm{M}(t)$ are eliminated, and $\mathrm{M}'(t)$ is formed.

(2) Structures are selected by using the Cross-generational Probabilistic Survival Selection (CPSS) method, and $\mathrm{P}(t)$ is formed from the structure with the best fitness value

```
begin
  t=0;
  initialize population P(t);
  evaluate structures of 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 of C(t);
    merge structures of 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);
      (2) select structures with CPSS method;
      (3) if the number of selected structures is
          smaller than N, introduce new structures;
  end;
end;
```

Fig. 1 Skeleton of DCGA.

in $M'(t)$ and the selected structures. In the CPSS method, structures are selected by using 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 the candidate structure and the structure with the best fitness value, L is the length of the string representing the structure, c is the shape coefficient, and α is the exponent. If the generated random number is smaller than p_s calculated for a 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 $M'(t)$ except the structure with the best fitness value.

(3) If the number of the structures in resultant $P(t)$ is smaller than N, then new structures generated by using random numbers are introduced by the difference between these numbers. In the traveling salesman problem, the hamming distance is calculated by considering only the order of the city in the structure, because the position of the city does not have a meaning.

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

Side-effect of crossover and mutation may destroy the better-performing schemata obtained so far. In the 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 and thus 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 in the examples mentioned later. On the other hand, in the simple GA, mutation is a background operator that ensures that the crossover has full range alleles so that the adaptive plan is not trapped on a local optimum [10], 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, as will be shown in later examples.

Equation (1) represents a curve that intersects the two points $[h = 0,\ p_s = c^{\alpha}]$ and $[h = L,\ p_s = 1.0]$ as shown in **Fig. 2**. The curvature of the curve is larger in the region of smaller h, whereas it becomes almost a straight line in the region of larger h. As α becomes smaller, the curvature in the region of smaller h becomes larger. When α is equal to 1, it becomes a straight line. As c becomes larger, p_s becomes larger and the curve approaches a horizontal straight line. Preliminary experiments on functions for the selection probability showed that the performance is closely related to the shape of the curve in the region of smaller h. Equation (1) was selected because it can simply and flexibly express various curves in the region of smaller h.

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 and an increase in the number of structures similar to it may result in premature convergence. Therefore, the selection of structures based on Eq. (1) is biased toward selecting structures with larger hamming distances from

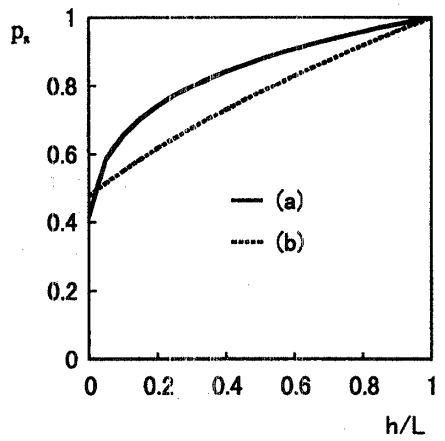

Fig. 2 Example curves of Eq. (1). (a) $\alpha = 0.19$, $c = 0.01$; (b) $\alpha = 0.51$, $c = 0.235$.

the structure with the best fitness value. The larger bias produces greater diversity of structures in the population.

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 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. I believe that the diversity of the structures in the population should be externally adjusted 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.

The selection with the CPSS method may slow the convergence speed to the global optimum, whereas it can be compensated and in fact improved by preventing the solution from getting stuck at a local optimum and stagnating. In this sense, the speed of convergence to the global optimum depends upon the diversity of structures in the population. In the DCGA, it can be controlled indirectly by the user through the constants α and c of Eq. (1).

The selection process uses the selection probability p_s, and 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. In the PES method[5], because the structures are deterministically selected in order of their fitness values, 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 PES method will be demonstrated later.

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 the structure is represented by a bit string, binary coding or gray coding[11] is usually used. With gray coding, the hamming distance between two structures can more accurately represent the degree of their similarity in the phenotype represented by decimal numbers. With the DCGA, because the performance with gray coding is superior to that with binary coding, as will be demonstrated later, it is recommended that gray coding be used.

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

In the DCGA, the diversity of the structures in the population is maintained by applying a large mutation rate, eliminating duplicate structures and the selection with the CPSS method. In addition, when the diversity is lost, it can automatically be restored by introducing new structures. 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 that may contain a schema concerning the global optimum 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. This mechanism is similar to that of simulated

annealing (SA), which can escape from a local optimum by accepting a solution based on a probability whose performance is worse than the present solution. In the DCGA also, the solution can escape from a local optimum in a similar way to the SA.

With the simple GA, better-performing structures can produce multiple offspring. Therefore, schemata for a dominating local optimum can increase rapidly and eventually dominate the population. On the other hand, with the 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 a structure (especially the structure with the best fitness value) from contributing many offspring to the next generation, and can eventually result in avoiding premature convergence.

The time complexity of the simple GA and the DCGA can be compared as follows. The number of function evaluations in the reproduction selection stage is N times per generation in both methods. The number of computations for the selection for reproduction is almost the same in both methods. In the survival selection stage, the number of computations for the DCGA is much larger than that for the simple GA because of the number of computations for the processes (1), (2), and (3) in Fig. 1. The time complexity of the check for the identity of structures in the process (1) is not so large, because the check needs to be performed among structures with the same fitness value. The time complexity of the calculation of the distance between the candidate structure and the structure with the best fitness value is $O(N)$. With the CHC[5], it is also $O(N)$. With the sharing function method[3], it is $O(N^2)$. Therefore, the computational cost of maintaining the diversity of structures is much lower for the DCGA than for the sharing function method.

With the simple GA, the parameters to be tuned are N, p_m, and p_c, whereas with the DCGA, they are N, p_m, c and α.

The originality of the DCGA lies in presenting a new genetic algorithm in the generation-replacement-type GA that combines the following ideas and in experimentally proving their effectiveness for attaining the global optimum. In the selection for survival, (1) duplicate structures are completely eliminated, and (2) structures for the next generation are selected on the basis of Eq. (1). The DCGA has a salient feature that the diversity of structures in the population (therefore the speed of convergence to the global optimum) can be externally controlled through the constants α and c in Eq. (1) so as to be in an appropriate condition according to the given problem. As far as I know, the CPSS method defined by Eq. (1) has not been presented in previous research.

The major difference between the DCGA and the CHC[5] is that the former employs the CPSS method and the latter the PES method. In terms of the approach to controlling diversity, the latter uses the self-adaptive difference threshold for mating, whereas the former is based on the externally adjusted selection probability.

If we compare the DCGA with the sharing function method[3], 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 method of the latter follows the paradigm of the traditional GA, and modifies the reproduction probability of a member of the population by adjusting the fitness value of the structures according to how many population members occupy a niche of the solution space.

4. Experimental Results

4.1 Computational Conditions

The performance of the DCGA has been tested on various benchmark problems and compared with those of the simple GA and existing algorithms. I present here the results for three standard benchmark problems that are difficult for GAs to optimize: deceptive functions[5], multimodal function f6[12], and 30-city traveling salesman problem (TSP)[13]. The f6 function is resistant to hill-climbing, nonseparable, and nonlinear, and thus satisfies the guidelines for test problems proposed by Whitley, et al.[14]. These problems were selected because the results reported in the literature for promising previous methods such as CHC[5),18] can be used for comparison.

Each problem has only one global optimum, which was searched by GAs. For both the DCGA and the simple GA, two-point crossover was used. For the DCGA, crossover was al-

ways applied to each pair in $P'(t-1)$. For the simple GA, the performances of the following three methods were compared: (1) the roulette selection method using the elitist strategy with or without fitness scaling, (2) the pure selection method with the fitness scaling proposed by Grefenstette [15], and (3) the stochastic remainder selection method without replacement [4),16)] with or without fitness scaling. The results of the first method without fitness scaling (De Jong's standard GA) [15], which showed the best performance, are described below. For the DCGA, the following three cases of computation conditions were tested in order to examine the effects of the methods (1) and (2) in Fig. 1 that are employed in the survival selection. Case-1: noneliminating duplicate structures and the PES method. Case-2: eliminating duplicate structures and the PES method. Case-3: eliminating duplicate structures and the CPSS method. Case-3 is the DCGA itself. With the PES method, N structures for the next generation are deterministically selected from $M(t)$ or $M'(t)$ in order of their fitness values in the process (2) in Fig. 1.

I examined combinations of best-performing parameter values, including the population size, changing their values little by little. I performed 50 trials per parameter set, changing seed values for the random number generator to initialize the population. The same 50 seed values were used for the trials with each parameter set. The trial was continued until the global optimum was attained by at least one structure (I call this convergence) or until the maximum number of function evaluation times ($MXFE$) was reached. The maximum number of function evaluation times was 50,000. The performance was evaluated by the rate of instances out of the 50 trials in which the GA converged (CVR) and the average number of function evaluation times in those trials that converged ($AVFE$). (An algorithm performs better on a function if it converges more often, or if it converges the same number of times as its competitor but in fewer evaluations.).

In the following, I present the best result obtained in each case. **Table 1** shows the definitions of major symbols used in the subsequent tables.

4.2 Deceptive Functions

Goldberg's order-3 deceptive functions [5)] were used. Their structure consists of a 30-bit binary string, and the value of the function is the sum of ten 3-bit subfunctions. The subfunctions are defined as in **Table 2**. The tightly ordered and loosely ordered functions were tested. The global maximum is 300.

Tables 3 and **4** show the best results for the tightly ordered function on the simple GA and the DCGA, respectively. **Table 5** shows the best result for the loosely ordered function on the simple GA with an $MXFE$ of 100,000, because no trials converged after 50,000 evaluations. **Table 6** shows the best results for the loosely ordered function on the DCGA.

4.3 Multimodal Function

The f6 function [12)] is as follows:

Table 1 Definitions of major symbols in Tables.

Symbol	Definition
G	Gray coding
B	Binary coding
N	Population size
p_m	Mutation rate
p_i	Inversion rate
p_c	Crossover rate
α	Exponent for probability function, Eq.(1)
c	Shape coefficient for probability function, Eq.(1)
CVR	Convergence rate (rate of successful trials)
$AVFE$	Average number of function evaluation times
$SDFE$	Standard deviation of function evaluation times
$AVEL$	Average value of error logarithm (EL)
$AVBF$	Average value of best fitness values
$MXFE$	Maximum function evaluation times

Table 2 Goldberg's order-3 deceptive function.

f(000)=28	f(001)=26	f(010)=22	f(011)=0
f(100)=14	f(101)=0	f(110)=0	f(111)=30

Table 3 Best result for tightly ordered deceptive function on simple GA.

N	p_m	p_c	CVR	$AVFE$	$SDFE$	$AVBF$
80	0.003	0.62	0.1	34720	8760	294.4

Table 4 Best results for tightly ordered deceptive function on DCGA.

Case	N	p_m	α	c	CVR	$AVFE$	$SDFE$	$AVBF$
1	84	0.095	—	—	0.98	19104	7078	300.0
2	100	0.036	—	—	1.0	8322	6154	300.0
3	4	0.008	0.51	0.33	1.0	6182	3452	300.0

Table 5 Best result for loosely ordered deceptive function on simple GA.

N	p_m	p_c	CVR	$AVFE$	$SDFE$	$AVBF$
110	0.0034	0.6	0.4	74591	18294	297.9

Table 6 Best results for loosely ordered deceptive function on DCGA.

Case	N	p_m	α	c	CVR	$AVFE$	$SDFE$	$AVBF$
1	8	0.086	—	—	0.7	31996	9284	299.2
2	4	0.08	—	—	0.98	24687	10207	300.0
3	4	0.045	0.37	0.83	1.0	14996	6512	300.0

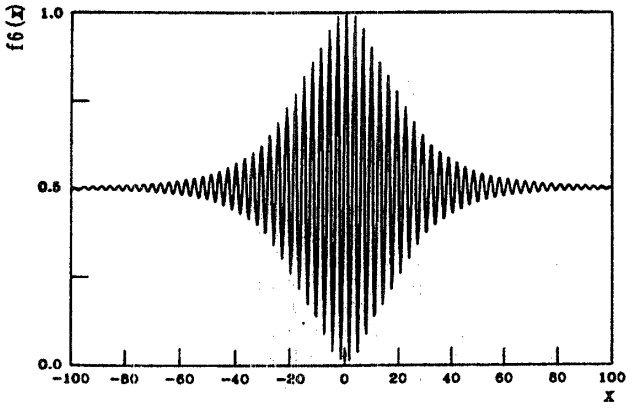

Fig. 3 Section of f6 function.

Table 7 Best results for f6 function on simple GA.

Code	N	p_m	p_c	CVR	$AVFE$	$SDFE$	$AVEL$
G	70	0.001	0.6	0.22	28280	7382	3.53
B	48	0.0009	0.6	0.28	14983	12328	3.89

Table 8 Best results for f6 function on DCGA.

Case	Code	N	p_m	α	c	CVR	$AVFE$	$SDFE$	$AVEL$
1	G	92	0.091	—	—	1.0	25994	6610	8.94
2	G	94	0.081	—	—	1.0	24145	5571	8.94
3	G	12	0.014	0.51	0.235	1.0	17795	9200	8.94
	B	12	0.014	0.51	0.235	0.66	22383	10317	7.40

$$\mathrm{f6} = 0.5 + \frac{0.5 - \sin^2\sqrt{x^2+y^2}}{[1+0.001(x^2+y^2)]^2}. \qquad (2)$$

This function is cylindrically symmetric about the z(f6) axis, and has a maximum value of 1.0 at the origin. **Figure 3** shows a section for $y = 0$ including the x and z axes. The points in the search space were coded as Cartesian x and y values in the range -100 to $+100$ with 22-bit code, respectively. Gray coding and binary coding were compared. The error of the best fitness values ($f_{\max}$) obtained in each trial is calculated by using the following equation:

$$EL = -\log_{10}(1 - f_{\max}) \qquad (3)$$

For the exact global optimum to be searched by GAs in the discrete space, $EL \cong 8.94$.

Tables 7 and **8** show the best results for the simple GA and the DCGA, respectively.

4.4 Traveling Salesman Problem

The Euclidean symmetric 30-city TSP [13] whose global optimum is 420 was tested. The structure was expressed by path representation. For both the DCGA and the simple GA, order crossover [13] was used. For the mutation method, the performance with order-based mutation [8], positionbased mutation, and inversion was compared. The results for the last of these which showed the best performance, are described below.

Table 9 Best result for 30-city TSP on simple GA.

N	p_l	p_c	CVR	$AVFE$	$SDFE$	$AVBF$
56	0.09	0.61	0.26	80132	14207	426.9

Table 10 Best results for 30-city TSP on DCGA.

Case	N	p_l	α	c	CVR	$AVFE$	$SDFE$	$AVBF$
1	110	0.206	—	—	0.52	20299	8596	421.8
2	120	0.196	—	—	0.92	21243	6748	420.2
3	22	0.098	0.19	0.01	0.94	23209	10578	420.1

Table 9 shows the best result for the simple GA with an $MXFE$ of 100,000, because no trials converged after 50,000 evaluations. **Table 10** shows the best results for the DCGA.

4.5 Summary of the Results

The following points were experimentally confirmed through investigation of the above three problems.

In all these three problems, the performance of the DCGA was remarkably superior to that of the simple GA.

With the deceptive functions and the 30-city TSP, the performance in case-2 was remarkably better than that in case-1. With the f6 function, however, the difference was slight. Therefore, it is obvious that although the degree of the effect is different for each problem, eliminating duplicate structures is effective for improving the performance.

With the deceptive functions and the f6 function, the performance in case-3 was remarkably better than that in case-2. With the 30-city TSP, however, the difference was slight. Therefore, it is obvious that although the degree of the effect is different for each problem, the CPSS method is superior to the PES method.

According to the results for the f6 function on the DCGA, the performance with gray coding is remarkably superior to that with binary coding. Therefore, it is recommended that gray coding be used for the DCGA.

The best results in case-3 obviously show 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 value of p_{s0} in the best results is 0.57 for the tightly ordered deceptive function, 0.93 for the loosely ordered deceptive function, 0.48 for the f6 function, and 0.42 for the 30-city TSP.

The best results in case-3 obviously show that there exists an optimum population size. It seems that a harder problem requires a larger

Table 11 Best results for each of the problems on CHC.

Problem	*N*	*MXFE*	*CVR*	*AVFE*	*SDFE*	*AVBF*
Oder-3 Dec.	50	50000	1.0	20960	980	300.0
f6 function			1.0	6496	725	—
30-city TSP			0.58	24866	2404	429.2

Table 12 Best results for each of the problems on Srinivas's method.

Problem	*N*	*MXFE*	*CVR*	*AVFE*
Oder-3 Dec.	100	20000	0.7	10533
f6 function	100	20000	0.8	10656
30-city TSP	1000	100000	0.7	—

population size. It should be noted that with the DCGA, the optimum population size is extremely small and good performance is obtained with such a small population. This indicates that if the structures are appropriately distributed in the solution space, only a small number of such structures are needed, and similar structures are harmful for reaching the global optimum.

According to the above, the DCGA seems to be especially suitable for the TSP and problems in which the global optimum is isolated as a deceptive function.

5. Discussion

It is interesting how well the DCGA performs in comparison with the leading existing methods. In previous research, however, the computational conditions varied from case to case, and some computational conditions and results were not described. Therefore, although exact comparisons of the performance are impossible, the best results are described in the following for the purpose of conjecturing as to the differences in the performance.

Table 11 shows the computational conditions and the best results in 50 trials using CHC for each of the problems[5),18)]. According to these results, the DCGA remarkably outperforms the CHC for the deceptive functions and the 30-city TSP. However, for the f6 function, the latter is remarkably superior to the former.

Table 12 shows the computational conditions and the best results when Srinivas's method was used for each of the problems[4)]. The numbers of trials was 30 for the first two problems and 10 for the last problem. The kind of order-3 deceptive function used was not described. The string length of the order-3 deceptive function was 15. With the f6 function, an approximate global optimum of 0.999 was used instead of the exact global optimum. In the 30-city TSP, the DCGA outperforms Srinivas's method.

In terms of Whitley's method, the performance described in the literature is as follows. In the f6 function using the population size of 100, the number of 9's below the floating point almost presents a peak when the number of function evaluation times is 4000 and the maximum value is smaller than 4[17)]. In the 30-city TSP using a very large population size of 1000 and an *MXFE* of 30,000, all the 30 trials converged[19)].

It should be noted that the optimum population size in the DCGA is extremely small in comparison with that in these previous methods. The salient feature of the DCGA is that good performance is obtained with such a small population size. In applications to practical large-scale problems, the combination of a GA and a local search algorithm is effective and efficient. In such an algorithm, the DCGA has the great advantage that the small population size can reduce the computational time consumed by applying the local search algorithm to each structure in the population.

6. Conclusions

A salient feature of the DCGA is that the diversity of structures in the population can be externally controlled so as to be in an appropriate condition according to the given problem. Within the range of the above experiments on standard benchmark problems, the following conclusions can be drawn. The methods employed in the DCGA are effective for attaining the global optimum. The optimum population size is extremely small and good performance is obtained with such a small population. The performance of the DCGA is remarkably superior to that of the simple GA. The DCGA may be a promising competitor to the GAs proposed in previous research. However, further evaluation of the DCGA in other complicated benchmark problems is required before firm conclusions may be drawn. In additional, theoretical analysis of the convergence process of the DCGA is required.

References

1) Baker, J.E.: Adaptive Selection Methods for Genetic Algorithms, *Proc. of an International Conference on Genetic Algorithms and Their Applications*, pp.101–111, Lawrence Erlbaum (1985).

2) Booker, L.: Improving Search in Genetic Algorithms, *Genetic Algorithms and Simulated Annealing*, pp.61–73, Morgan Kaufmann (1987).
3) Goldberg, D.E., et al.: Genetic Algorithms with Sharing for Multimodal Function Optimization, *Proc. Second International Conference on Genetic Algorithms*, pp.41–49, Lawrence Erlbaum (1987).
4) Srinivas, M., et al.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, *IEEE Trans. Systems, Man and Cybernetics*, Vol.24, No.4, pp.656–667 (1994).
5) Eshelman, L.J.: The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, *Foundation of Genetic Algorithms*, pp.265–283, Morgan Kaufmann (1991).
6) Greene, J.R.: A Role for Simple, Robust 'Black-box' Optimisers in the Evolution of Engineering Systems and Artifacts, *Proc. Second IEE/IEEE Int. Conference on Genetic Algorithms in Engineering Systems*, pp.427–432 (1997).
7) Whitley, D., et al.: Test Driving Three 1995 Genetic Algorithms: New Test Functions and Geometric Matching, *Journal of Heuristics*, Vol.1, pp.77–104 (1996).
8) Shimodaira, H.: DCGA: A Diversity Control Oriented Genetic Algorithm, *Proc. Second IEE/IEEE Int. Conference on Genetic Algorithms in Engineering Systems*, pp.444–449 (1997).
9) Shimodaira, H.: DCGA: A Diversity Control Oriented Genetic Algorithm, *Proc. Ninth IEEE Int. Conference on Tools with Artificial Intelligence*, pp.367–374 (1997). http://www.hi-ho.ne.jp/shimo-hi/new_ga.htm
10) Holland, J. H.: *Adaptation in Natural and Artificial Systems*, p.111, MIT Press (1992).
11) Caruana, R., et al.: Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms, *Proc. 5th Int. Conference on Machine Learning*, pp.153–161, Morgan Kaufman (1988).
12) Schaffer, J.D., et al.: A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, *Proc. Third International Conference on Genetic Algorithms*, pp.51–60, Morgan Kaufmann (1989).
13) Oliver, I.M., et al.: A Study of Permutation Crossover Operators on the Traveling Salesman Problem, *Proc. Second International Conference on Genetic Algorithms*, pp.224–230, Lawrence Erlbaum (1987).
14) Whitley, D., et al.: Building Better Test Functions, *Proc. Sixth International Conference on Genetic Algorithms*, pp.239–246, Morgan Kaufmann (1995).
15) Grefenstette, J.J.: Optimization of Control Parameters for Genetic Algorithms, *IEEE Trans. Systems, Man, and Cybernetics*, Vol.SMC-16, No.1, pp.122–128 (1986).
16) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization & Machine Learning*, p.121, Addison Wesley (1989).
17) Davis, L.: *Handbook of Genetic Algorithms*, p.47, Van Nostrand Reinhold (1991).
18) Eshelman, L.J., et al.: Real-Coded Genetic Algorithms and Interval-Schemata, *Foundations of Genetic Algorithms 2*, pp.187–202, Morgan Kaufmann (1993).
19) Starkweather, T., et al.: A Comparison of Genetic Sequencing Operators, *Proc. Fourth International Conference on Genetic Algorithms*, pp.69–76, Morgan Kaufmann (1991).

(Received April 7, 1998)
(Accepted March 5, 1999)

Hisashi Shimodaira received the B.E. and the M.E. degrees in 1969 and 1971, respectively, from Tokyo Metropolitan University. Then, he was employed by NTT and did research on development of computer software at Musashino telecommunication laboratory. Through the research in NTT, he received the Doctor of Engineering Degree in 1982 from Tokyo Metropolitan University. He is currently a professor in Department of Information and Communication, Bunkyo University, Japan. His research interests are in the area of artificial intelligence, computational intelligence, image processing, signal processing, multimedia processing, and communication technologies. He is a member of IEICE, JSAI and IEEE.