Distributed Genetic Algorithms with a New Sharing Approach in Multiobjective Optimization Problems

Tomoyuki HIROYASU **Mitsunori MIKI** **Sinya WATANABE**

Doshisha University, Dept. of Knowledge Engineering and Computer Sciences
1-3 Tatara Miyakodani, Kyotanabe
Kyoto 610-0321, Japan

tomo@is.doshisha.ac.jp mmiki@mail.doshisha.ac.jp sin@mikilab.doshisha.ac.jp

[1] **Abstract- In this paper, a new distributed genetic algorithm for multiobjective optimization problems is proposed. In this approach, the island model is used with a distributed genetic algorithm and an operation of sharing for Pareto-optimum solutions is performed with the total population. In multiobjective optimization problems, the Pareto-optimum solutions should be derived for designers. Because the Pareto-optimum solutions are the set of optimum solutions that are in the relationship of trade-off, not only the accuracy but also the diversity of the solutions should be high. The effect of the distributed populations leads to the high accuracy and the sharing effect leads to the high diversity of solutions. These effects are examined and discussed through some numerical examples that have more than three objective functions.**

1 Introduction

In real world problems, there are usually multiple objective functions that are in the relationship of trade-off. Those optimization problems are called multiobjective optimization problems. In portfolio problems, for example, when the safety increases, the return decreases. On the other hand, when the return increases, the safety decreases. In this case, the safety and return are objective functions and those are in the relationship of trade-off.

The genetic algorithm is one of the most powerful optimization methods based on the mechanics of natural evolution (gold89). This algorithm can be used in the discrete design field and it is said that this algorithm can find the global optimum even when there are several local optima. Therefore genetic algorithms are very useful, but there are some problems. One of the problems is the necessity of the huge number of iterations. Hence, to solve practical problems by genetic algorithms, genetic algorithms should be speeded up somehow. One of the solutions is the parallelization of genetic algorithms. There are also several ways to perform genetic algorithms in parallel, but one of the ways is so called the distributed genetic algorithms. In the distributed genetic algorithms, the entire population is divided into small groups which is called sub populations or islands. It is reported that distributed genetic algorithms have some advantages compared to genetic algorithms with single populations (miki98).

This distributed genetic algorithms can also be expand to solve multiobjective optimization problems. There are some researches that are focused on the genetic algorithms in multiobjective optimization problems (haje92, fons93, fons95). On the other hand, there are very few researches focused on the distributed genetic algorithms in multiobjective optimization problems. Especially, the efficiency of the sharing which is an operation that should be performed to keep the diversity of the solutions in genetic algorithms for multiobjective optimization problems is not well discussed.

In this study, a distributed genetic algorithm is examined in multiobjective optimization problems. Especially, a new model of distributed genetic algorithms is proposed. In this model, the operation of sharing is used for not only the population of the island but also for the entire population. To examine the effects of distribution and sharing, this algorithm is implemented with one processor. The effectiveness of the proposed method is discussed through the numerical examples that have more than three objective functions.

2 Genetic Algorithms in Multiobjective Optimization

Problems that have some multiple objective functions can be called multiobjective optimization problems (MOPs). In MOPs, to derive Pareto-optimal solutions (fons95) is one of the goals. Pareto-optimal solutions are the set of optimum solutions that are in the relationship of trade-off. There exist several algorithms that find Pareto-optimal solutions. Genetic algorithms (GA) is one of those methods.

There are some studies concerned with genetic algo-

[1]Proceedings of Congress on Evolutionary Computation, Vol. 1, 1999

rithms in multiobjective optimization problems. Fonseca and Fleming reviewed evolutionary algorithms in multiobjective optimization (bent80) and they also well summarized the studies on genetic algorithms in multiobjective optimization problems (fons98). In single-objective optimization problems, there is only one optimum point and GAs have disadvantages compared to classical optimization algorithms with respect to the number of iterations because GAs are multi-point searching algorithms. On the other hand, in MOPs, the solutions are set of multi-points and it can be said that GAs are suitable for MOPs.

At first, in GAs for MOPs, population is scattered in the design field or the objective field like Figure 1. Parents are selected to generate children. New population is selected somehow and the frontier moves toward the Pareto-optimum solutions.

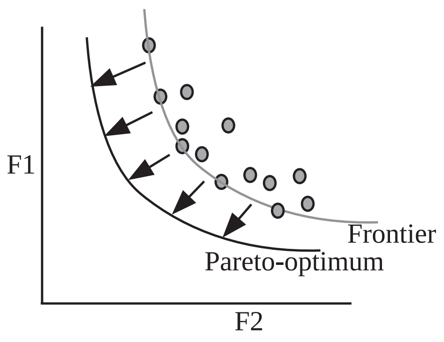

Figure 1: Pareto-optimum solutions

The GAs for MOPs roughly divided into two categories from the selection point of view. Those are the non-Pareto ranking selection and the Pareto ranking selection, respectively.

Schaffer proposed the vector evaluated genetic algorithm (VEGA) in MOPs (scha85). He is probably the first to expand GAs in MOP. In VEGA, the population is divided into sub-populations. In each sub-population, only one objective function is evaluated and the children are selected according to this objective function. This algorithm is very easy to apply in parallel. However, the solutions tend to concentrate on one point where one of the values of the objective functions is high but others are very low. The fact that the Pareto optimality is not treated explicitly in VEGA leads that uniformly distributed Pareto-optimum solutions can not be obtained.

Pareto ranking selection is treated the Pareto optimality of solutions explicitly. Goldberg is determined the ranking as follows (gold89). At first, all population is set their ranking r=1. Secondly, Pareto-optimum solutions are chosen from the population and these are removed from the population. Then, the ranking is renumbered r=r+1. This routine is continued until the rankings of all population are fixed.

Usually, these rankings are used as the fitness function for a selection such as the roulette selection. There are several types of fitness functions based on the Pareto ranking. Fonseca proposed the fitness functions that consist of not only rankings but of also the number of subordinated populations (fons93). Horn is proposed the way to construct the fitness functions with these rankings and sharing explained in the next section. In this study, only rank-1 populations are chosen and all rank-1 populations remain in the next generation. It takes much time when the population whose rank is not 1 are considered. Since crossover is performed in the design variable fields, the diversity is maintained adequately. This is a kind of elite selection of GAs in single-objective optimization problems.

In a simple genetic algorithm, the size of population is fixed. In multiobjective genetic algorithms, the number of populations increases with the generations since the frontier also expands with the generations. To prevent the explosion of the number of solutions, the sharing techniques should be introduced in this approach.

For designers, it is not useful when the Pareto-optimum solutions are concentrated on one point. Therefore, the Pareto-optimum solutions that have high diversity should be obtained. The sharing (horn94) is the operation to scatter the solutions over the Pareto-optimum solution set.

Usually, the sharing function $s(d)$ is defined with the distance $d(x_i, x_j)$ between two individuals i and j as follows,

$$s(d) = max\left\{0, 1 - \frac{d}{\sigma_{share}}\right\}. \quad (1)$$

Where σ_{share} is the parameter to share the solutions and called a sharing radius. Then the values of the fitness function is changed with the sharing function.

In this study, the sharing does not change the fitness function but reduces the size of population directly, like Figure 2. Therefore, only one population remains within the sharing radius.

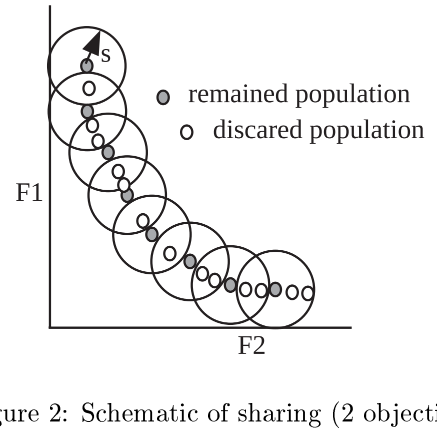

Figure 2: Schematic of sharing (2 objectives)

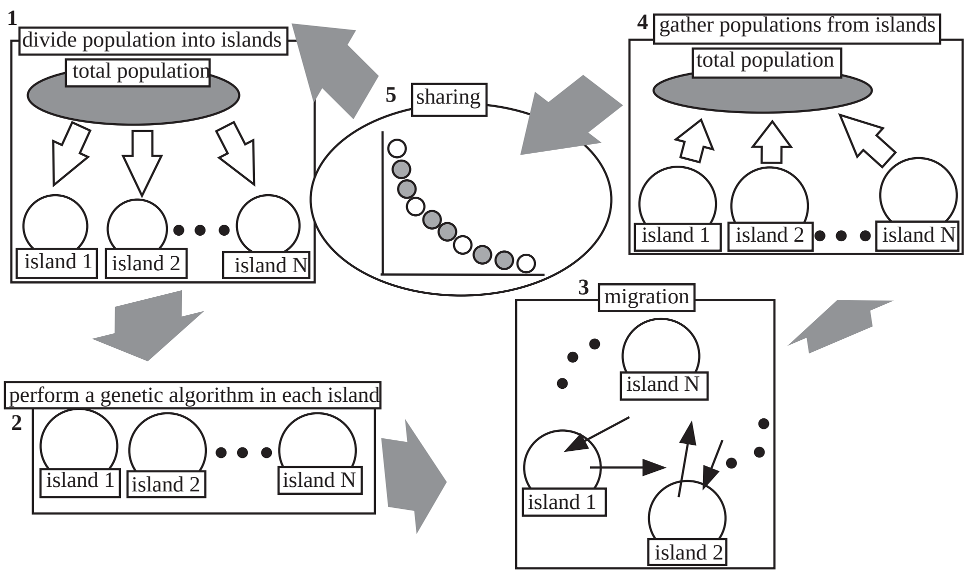

Figure 3: Distributed genetic algorithms with sharing

Because GAs have intrinsic parallelism (gold89), it is easy to perform GAs in parallel. The ways of parallization of genetic algorithms is roughly divided into two ways. One of them is to parallelize evaluations and the other is to parallelize population. In the way of parallelization of population, the population is divided into small groups that are called islands and this approach is called distributed genetic algorithms (DGAs). A genetic algorithm is carried out in each island separately. After some generations, some individuals in each island are chosen and sent to another island. This operation is called migration. The migration interval is a parameter in DGAs. The number of individuals that migrate to another islands is another parameter that is called a migration rate. There are a lot of studies concerned with distributed genetic algorithms. It was made clear that DGAs can find the optimum with small population size and short calculation times (tane89, star91, muhl91).

DGAs can be easily expanded to the application in MOPs. In this case, DGAs are also roughly divided into two approaches. Those are the VEGA approach and the island model approach. In the island model approach, the population is divided into sub-populations which are called islands. In the islands, ranking or non ranking selections are performed. After a certain generations, some individuals are chosen and are migrated to another islands.

There are few studies concerned with DGAs in MOPs. Hiyane examined the DGAs in MOPs and concluded that DGAs are powerful algorithms under parallel computations (hiya97). However, the diversity of solutions becomes low since the population size becomes small compared to GAs with a single population. Therefore, the sharing should be carried out for the total population to increase the diversity. A DGA with the sharing to the total population is the base idea of the proposed approach explained in the following chapter.

3 Distributed Genetic Algorithms with Sharing

In this study, a new approach of distributed genetic algorithms in multiobjective optimization problems is proposed. In this approach, the island model is taken for a distributed genetic algorithm and a sharing is performed when the number of the frontier solutions exceed the certain size. The concept is shown in Figure 3 and the flow of the algorithm is shown in Figure 4.

In distributed genetic algorithms in multiobjective optimization problems, the size of sub-population is smaller than that of canonical genetic algorithms. Therefore, the sharing does not take much time compared to the single island. This leads that objective functions can be called more. On the other hand, the diversity of the solutions are getting smaller. To increase the diversity, the sharing for the total population is necessary. The proposed algorithm is the combination of the advantages of distributed genetic algorithms and the sharing for the total population. At first, the total population is divided into several islands. Then, a simple genetic algorithm is performed in each island. After certain generations, the migration is carried out. During the distributed genetic algorithm, the size of the frontier solutions are increasing. When their size exceeds a criterion, the sharing is performed in the total populations. And again, the total population is divided into several

```
begin
  (initialization)
  determine
        the number of islands M
        he population size in one island N
        the number of sharing p
  divide population into islands
  k = 0

  (generation k)
  while convergence condition do
        i=0

        (in ith island)
        while i<M do
                evaluaton
                crossover
                mutation
                sharing (OPTIONAL)
        end

        (migration)
        if migration condition then
                begin
                        migration
                end

        (sharing)
        if sharing condition then
                begin
                        gather population from islands
                        sharing
                        divide population into islands
                end

  end
end
```

Figure 4: Flow of distributed genetic algorithms with sharing

islands. This routine is continued until the convergence condition is satisfied. In this algorithm, the sharing in each island is not performed usually, but when it takes a lot of time to share the solutions in the total population, the sharing is also performed in each island.

It is supposed that the proposed algorithm has the following four advantages. At first, with this algorithm, an efficient sharing can be performed. Therefore, the solutions have high diversity. Secondly, when the sharing is performed only in each island, the size of sub-population is unbalanced in each island. This algorithm can correct the sizes of sub-populations. This means that load balance in parallel computing can be performed automatically. Thirdly, there is a possibility to remove the operation of migration. In this study, migration is performed. However, it is supposed that the division of the total population into multiple islands has also the effect of migration. Finally, the setting of the sharing parameter in the entire design space is much easier than that of the sharing in islands since it is difficult to grasp the number of frontier solutions in each island.

The proposed algorithms is examined and discussed with numerical examples in the following chapters.

4 Numerical Examples

4.1 Function Problems

To examine the performance of the proposed approach, the following simple MOP is solved and the Pareto-optimum solutions are derived in the next section.

Objective functions

$$f_i = -x_i \quad (i = 1, 2, \ldots, n) \tag{2}$$

Constraints

$$g_j = -x_j \quad (j = 1, 2, \ldots, n) \tag{3}$$

$$g_{n+k} = x_k - 6 \quad (k = 1, 2, \ldots, n) \tag{4}$$

$$g_{2n+1} = 1 - x_1 \cdot x_2 \cdot \ldots x_n \tag{5}$$

This MOP is very simple, but it is easy to expand to arbitrary N-dimension problems. In most previous studies there are only two objective functions. The MOP that has two objective functions is easy to demonstrate because it is easy to represent the Pareto-optimum solutions. Hence, it is easy to grasp the effect of the algorithms. However, the MOP which have more than 4 objective functions have another problems compared to the MOP which have only two objective functions. Those are the problems of the population size that needs to express the Pareto solutions and the problem of the difficulty of the sharing. Therefore, even when the algorithms that lead good solutions in the problem that has two objective functions, it sometimes happens that the algorithms do not work for the problems that has more than three objective functions. Because of this, it is necessary to test and discussed the algorithm in the MOP where there are more than three objective functions.

On the other hand, it is very difficult to grasp the Pareto-optimum solutions where there are more than 4 objective functions in MOPs because it is impossible to draw all of the Pareto-solutions at once. It is another problems to overcome and this is the future challenge of the MOPs.

As it is mentioned before, to examine the effect of the algorithms, the real Pareto-optimum solutions should be known. In this numerical example, it is very easy to find the real Pareto-optimum solutions and it is also easy to get the high accuracy of the solutions. In Figure 5, the Pareto-optimum solutions are shown in case of two and three objective functions, respectively.

4.2 Genetic Algorithm in Each Island

In this study, genetic operators are performed in each island. Those are crossover and selection. In this computation, the mutation is not performed and crossover rate is 1. Because this MOP is in real value space, the real value vectors are used as coding instead of binary

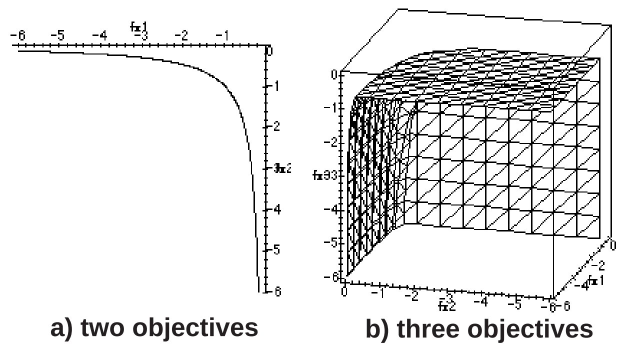

Figure 5: Pareto-optimum solutions (2 and 3 objectives)

coding. Because of this coding, conventional crossover operations can not be used. Here the normal distribution crossover is used as follows. When there are n design variables, n+1 points are selected probabilistically. The center of gravity G of n+1 points is derived. Then the child is derived the following equation.

$$\overrightarrow{\mathrm{C}} = \overrightarrow{\mathrm{G}} + \sum_{i=1}^{N} N(0, \sigma_i^2)\overrightarrow{\mathrm{GP_i}} \tag{6}$$

Example of the crossover where there are two objectives are shown in Figure 6.

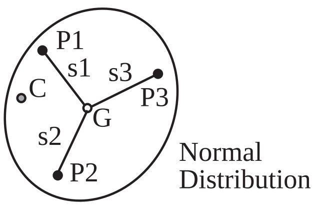

Figure 6: Normal distribution crossover

When a new child break the constraints, another child is derived using the following equation and shown in Figure 7.

$$\overrightarrow{\mathrm{C_{new}}} = \overrightarrow{\mathrm{P_1}} + \alpha\overrightarrow{\mathrm{P_1C_1}} \tag{7}$$

In this equation, α is reduced until the child satisfies constraints. Only rank 1 solutions of the frontier solutions are chosen when the selections. The size of frontier solutions is not limited in this numerical examples.

4.3 Evaluation methods of algorithms in multiobjective optimization problems

In this study, the algorithms are evaluated from the following 4 points of view. Those are the number of solutions, error, cover rate, and coefficient of variation.

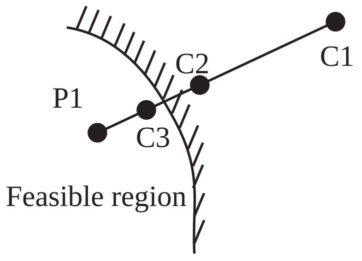

Figure 7: Generation of child satisfies the constraints

The number of solutions
It is said that the size of population is one of most important index for the performance of algorithms in MOPs. Especially, the size of population of rank 1 is very important. If the algorithms can not find enough number of rank 1 solutions, designers can not grasp the relationship between the objective functions. In that case, it can be said that the algorithm cannot produce good solutions. On the other hand, an extra population increases the computational cost. Therefore, there should be enough number of solutions that have enough accuracy and enough cover rate.
Error
When the real Pareto-optimum solutions are known, the accuracy of the solutions is one of the important index for the Pareto optimum solutions. The error of solutions E can be derived as follows

$$E = \sqrt{\sum_{i=1}^{n}(d_i)^2/N} \tag{8}$$

where d is the distance from solution to the real Pareto-optimum solutions. It is obvious that the real Pareto-optimum solution should be known to derive the accuracy. Therefore it is rather difficult to use this as the index in real-world problems.
Cover rate of solutions
The index of error is not enough for evaluating the Pareto-optimum solutions of MOPs. For example, if there is only one Pareto-optimum solution. The accuracy is very high but it is not enough to express the Pareto-optimum solutions.

To support the disadvantage of the accuracy, a cover rate is proposed in this study. At fist, the maximum and minimum values of each object function are searched. The distances between maximum and minimum are divided by a certain number. The number of intervals where a solution exists is counted. Total number is averaged by the total number of intervals. This average number becomes the cover rate. When this index is close to 1, the solutions well cover from the maximum to the minimum of the Pareto optimum solutions. This is the only index for finding solutions covered the entire area or not. Therefore, this index should be used with the index of error.

Coefficient of variation
Ideally, the Pareto-optimum is shown with the smallest number of solutions. To find the extra solutions, the coefficient of variation is introduced in this study and it expresses the index of diversity. The coefficient of variation is derived as follows. At first, one of the solutions are focused on and the number of population that are in the certain distance is counted. Other solutions are also focused and the number of them are counted. Then, the average of standard deviations of this value becomes the coefficient of variation of each solution. Therefore, when the coefficient of variation becomes 1, the solution has high variety. ‘

5 Results and Discussions

In this study, the problem that is explained in sub section 4.1. This problem has 4 objective functions is solved. All results are the average of 10 trials. Migration is operated every other generation and 10% of each population of the island migrates to the other island randomly. These parameters affect to the solutions and the discussion about it is the future topics.

The proposed algorithm that is explained in section 3 can be performed on a parallel computer. However, to make the characteristics of the effects of the algorithm clearly, every numerical example is performed on a single CPU computer.

5.1 Distributed Effect

In Table 1, the results of distributed the genetic algorithm (DGA, island model) and the canonical genetic algorithm (CGA, 1 island) is shown. In the both cases, total initial population size is 1000, and the calculation is terminated when the evaluation function is called more than 1000 times. In the CGA, the population size is reduced about 80 by sharing when the size exceeds 2500. In the DGA, there are 10 islands and the sharing is performed only in the islands. This means that the sharing is not performed to total population in DGA. The population size of each island is reduced by about 50 by the sharing when the size exceeds 250.

Table 1: Effect of distribution

	number of solutions	error	cover ratio	coefficient of variation	generations	calculation time [s]
1 island	1980	0.191	0.856	2.46	6	194.9
10 islands	2690	0.196	0.853	3.10	6	34.3

From this results, it is obvious that it takes much time in the CGA compared to the DGA. In the DGA, the size of population in island is not so big and it does not take much time in the sharing operation. On the other hand, the error and the cover ratio are almost the same. Therefore, when the calculation time is termination condition, the accuracy or the cover rate will increase in the DGA. On the other hand, diversity(coefficient of variation) is not so good in the DGA because each island has each frontier solutions and there exist the overlapped solutions.

5.2 Distributed Genetic Algorithm with sharing

It is clarified that the DGA derives the solutions that have high accuracy and high cover rate, but low diversity. When the diversity is low, it means that the solutions are gathered near one point. In MOPs, not only the accuracy of the solutions but also the high diversity is also important. The proposed algorithm shown in section 3 is one of the solutions to find the Pareto-optimum solutions that have high diversity.

In the numerical examples, the sharing reduces the total population size into 80 from 2500.

In Tables 2 and 3, the results of the proposed algorithm are shown. The results in Table 2 are derived when the calculation is terminated after 2000 times of the objective function call. The results in Table 3 are also derived when the calculation is terminated after a certain calculation time.

From Table 2, it can be seen that the DGA with sharing can find solutions that have high accuracy, high cover rate, and high diversity. The sharing performed in the proposed algorithm worked well to find good solutions. On the other hand, this sharing takes a lot of time. Table 3 points that the DGA with the sharing iterates only 3 generations. Therefore, it can be said that this solutions are not evolved well.

From these results, the DGA with the sharing can derive good solutions. Therefore, it has an advantage for the problems where it takes a lot of time to evaluate the objective function, such as structural optimization problems. However it has a disadvantage that the calculation time is very short.

Table 2: DGA and DGA with the sharing (termination condition = number of function call)

	number of solutions	error	cover ratio	coefficient of variation	generations	calculation time [s]
DGA	3888	0.171	0.855	4.11	8.7	91.0
DGA with sharing	3079	0.153	0.855	3.10	10.1	563.1

To make clear the advantage of the DGA with the sharing when the calculation time of the function evaluation is not short, function sleeps 0.001[s], 0.01[s], 0.05[s], or 0.1[s] when the function is called. In Figures 8 and 9, the error and cover ratio of solutions that are derived

Table 3: DGA and DGA with the sharing (termination condition = calculation time)

	number of solutions	error	cover ratio	coefficient of variation	generations	number of function call
DGA	3422	0.182	0.856	3.65	7.8	18998
DGA with sharing	1581	0.226	0.847	2.15	3.0	4985

by by the DGA and the DGA with the sharing are expressed when the sleep time of the function is changed respectively.

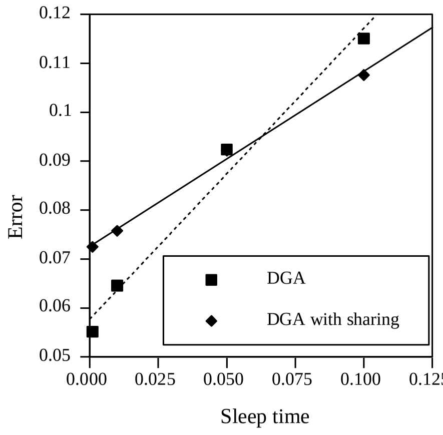

Figure 8: Error

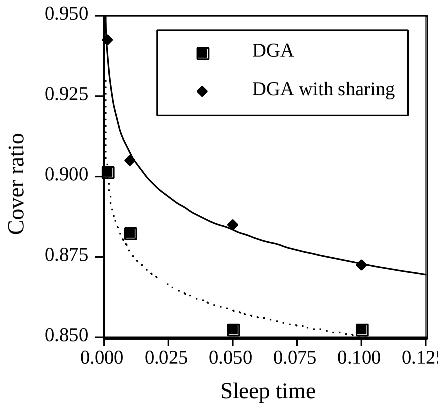

Figure 9: Cover ratio

From Figures 8 and 9, it is clarified that when the sleep time of the function is long, the quality of the Pareto-solutions that are derived by the DGA with the sharing are better than those derived by the DGA. The cover ratio of the DGA with the sharing is always better than that of the DGA. This result means that the proposed approach is useful when it takes much time to evaluate objective functions such as in structural design problems.

5.3 Sharing in both sub-population and total population

The proposed distributed genetic algorithm with the sharing takes a lot of time because of the sharing with total population when the calculation time of the objective functions is very short. To shorten the total calculation time, the sharing should be performed not only with the total population, but also in each island. In this case, the status in the islands are different, the sizes of the population of the islands are different each other. In this section, this type of sharing is called the hybrid sharing.

The results are shown in Table 4. In the hybrid sharing algorithm, the calculation time is shorter than that of DGA with the sharing. The diversity(coefficient of variation) is also made progress and this result shows the good effect of the sharing both in each island and with the total population. On the other hand, the accuracy of the solutions is not so good. This comes from the lack of the number of the solutions. Therefore, the parameter of sharing should be tuned up.

Table 4: DGA, DGA with sharing and Hybrid sharing

	number of solutions	error	cover ratio	coefficient of variation	generations	calculation time [s]
DGA	3888	0.171	0.855	4.11	8.7	91.0
DGA with sharing	3079	0.153	0.855	3.10	10.1	563.1
Hybrid sharing	2922	0.183	0.858	2.43	10.0	275.5

6 Conclusions

1. In this paper, a new method of distributed genetic algorithms for multiobjective optimization problems is proposed.

2. Indexes that can evaluate the performance of algorithms are introduced. Those are population size, error, cover rate, and coefficient of variation. These indexes can be applied to the problems that have more than three objectives.

3. A numerical example is shown to express the effectiveness of the proposed method. Conventionally, the multiobjective optimization problems with only two objectives. This example introduced here can be extended to arbitrary number of dimensions in the design space. At the same time, the accurate Pareto-optimum solutions can be derived easily.

4. In the proposed approach, the distributed genetic algorithm is performed with some sub-populations. When the frontier solutions exceed the criterion

number, the sharing is performed with the total population that is constructed from the subpopulations.

5. The proposed approach is verified by the numerical examples. It is clarified that the sharing with the total population increases the diversity and the accuracy of the solutions. This approach takes some time in the sharing, but this approach becomes useful when it takes much time to evaluate objective functions.

6. The algorithm where the sharing is performed in islands and in total population is also performed. This approach reduces the calculation time and makes some increase in the diversity, while the accuracy of the solutions is decreased.

Bibliography

[bent80] Ben-Tai, A. (1980) "Multiple Criteria Decision Making Theory and Application," Vol. 1777 of Lecture Notes in Economics and Mathematical Systems, pp. 1-11, Springer-Verlag

[fons93] Fonseca,C. M. and Fleming, P. J. (1993) "Genetic algorithms for multiobjective optimization," Proc. of 5th International Conference on Genetic Algorithms, pp. 416-423

[fons95] Fonseca, C. M. and Fleming, P. J. (1995) "An overview of evolutionary algorithms in multiobjective optimization," Evolutionary Computation 3(1), pp.1-16

[fons98] Fonseca, C. M. and Fleming, p. J. (1998) "Multiobjective optimization and multiple constraint handling with evolutionary algorithms part1," Trans. on Sys., man and cyber. Part A, Systems and Humans, 28(1), pp.26-37

[gold89] Goldberg, D. E. (1989) "Genetic Algorithms in search, optimization and machine learning," Addison-Wesly

[haje92] Hajela, P. and Lin, C. Y. (1992) "Genetic Search strategies in multicriterion optimal design," Structural Optimization 4, pp. 99-107

[hiya97] Hiyane, K. (1997) "Generation of a set of Pareto-optimal solutions for multiobjective optimization by parallel genetic algorithms and its quantitative evaluation," Proc. of No.9 automatic system symposium, pp. 295-300, in Japanese.

[horn94] Horn, J. Nafpliotis, N. and Goldberg, D. E. (1994) "A niched Pareto genetic algorithm for multiobjective optimization," Proc. of 1st IEEE International Conference on Evolutionary Computation, pp. 82-87

[miki98] Miki, M. (1998) "Parallel Genetic Algorithm with Parameter Free Approach," Proc. of the ICES 98, pp. 582-587, Vol. 1

[muhl91] Muhlenbein, H., Schomisch, M. and Born, J. (1991) "The parallel genetic algorithm as function optimizer," Proc. of 4th International Conference of Genetic Algorithms, pp.271-278

[scha85] Schaffer, J. D. (1985) "Multiple objective optimization with vector evaluated genetic algorithms," Proc. of 1st International Conference on Genetic Algorithms and Their Applications, pp. 93-100

[star91] Starkweather, T., Whitley, D., and Mathias, K. (1991) "Optimization using distributed genetic algorithms," Proc. of Parallel Problem from Nature, pp.176-184

[tane89] Tanese, R. (1989) "Distributed Genetic Algorithms" Proc. of 3rd international Conference on Genetic Algorithms, pp.432-439