Discussion of Parallel Model of Multi-Objective Genetic Algorithms on Heterogeneous Computational Resources

ABSTRACT

In this paper, a parallel model of Multi-objective Genetic Algorithms supporting a hetero calculation environment is discussed. In this parallel model, two individuals close each other in the objective space are sent to each calculation resource. Then, crossover is performed several times on each calculation resource and the number of offspring generated is changed dynamically adapting to the performance of the calculation resources. Here, this improved parallel model is compared with the original master-slave model to discuss the accuracy of the proposed model through computational experiments on heterogeneous computational resources. The results indicated that the proposed parallel model enabled utilization of the maximum performance of all calculation resources and has high search ability of Pareto-optimal solutions.

Keywords

Multi-objective optimization, Genetic algorithms, Parallelization

1. INTRODUCTION

The computational time needed to solve real-world multi-objective optimization problems is usually large. Therefore, it is very important to reduce the computational time with parallel processing. There have been a number of studies regarding parallel distributed implementation. For example, Deb *et al.* [1] proposed and discussed an approach that uses a distributed Genetic Algorithm(GA). Although there have been many studies of parallel evolutionary multiobjective optimization(EMO) [1, 2, 3, 4, 5, 6], the degree of parallelization is very small. On the other hand, the recent development of large clusters and grid computing, which have unified the calculation resources online, have made huge resources readily available for such computational tasks. To make use of these huge resources, it is necessary to consider the parallel model where many processes can be performed in parallel.

In this paper, we discuss the parallel model of Multi-objective Genetic Algorithms and propose a new parallel model supporting huge and heterogeneous computational resources. In the proposed model, many processes can be performed in parallel, and crossover operation and evaluation are performed on each process. Therefore, this approach can be applied on huge clusters and the Grid. At the same time, the number of offspring generated by crossover can be changed dynamically in this model. This mechanism is suitable for heterogeneous calculation environments, such as the Grid computational environment. In this paper, the performance of the proposed parallel model is compared with original master-slave model and the feasibility of the proposed model is discussed.

2. PARALLEL MULTI-OBJECTIVE GENETIC ALGORITHMS

Similarly to single-objective optimization studies, huge computational time is needed to solve real-world problems. Especially, the dimension of the true Pareto-optimal front increases when the number of objectives increases. Therefore, a large population size is required to reach a well-distributed Pareto-optimal front. This results in a huge computational time. One of the solutions to reduce the computational time is to perform EMO in parallel.

Many studies of parallel EMO have been performed, most of which have made use of the master-slave model or island model [1, 2, 3, 4, 5, 6]. In the master-slave model, one master processor runs the GA operations and slave processors are used for evaluation purposes only. In this model, when the number of processors P is used, the ideal acceleration is P. Any algorithm can be applied to this model and the obtained solutions would be equivalent to the solutions obtained with the original algorithms using a single processor. In the island model, a population is divided into a number of subpopulations and a processor is assigned per subpopulation. In this model, different EMOs are run on different processors and some solutions are migrated between processors after every few generations.

A good parallel EMO implementation was reported previously by Deb [1]. However, in this model, only a small degree of parallelization is assumed because this study used the island model. The master-slave model is effective to increase the degree of parallelization by at least the population size. On the other hand, many resources are becoming available due to the development of large clusters and grid

computing, which have unified online calculation resources. However, the resources on the Grid are heterogeneous, and it is therefore also necessary to consider a parallel model that can be applied to heterogeneous environments, such as grid computing. This paper discusses parallel EMO where the master-slave model is extended supporting a heterogeneous grid environment. The following two aspects should be kept in mind while proposing parallel EMO in grid computing:

- The resources in the grid environment differ in their performance. Therefore, it is necessary to take into consideration the parallel model corresponding to the heterogeneous grid environment.
- Overheads, such as communication times, must be sufficiently small as compared with evaluation time. As communication time in the grid environment becomes large as compared with parallel processing performed on a PC cluster, it is necessary to take into consideration the parallel model that can hide the overhead.

As the calculation resources in the grid environment have differences in performance, when all calculation resources have the same number of individuals to be evaluated, the calculation resources with inferior performance would require more time for evaluation, and would act as a bottleneck to progression to the next generation. Therefore, it is necessary to distribute the tasks adapted for the calculation resources. Here, we propose a parallel technique that improves on the master-slave model. Each slave process not only evaluates, but also performs crossover. Then, the number of offspring generated by the crossover operation is changed dynamically adapting to the performance of the calculation resources. Moreover, we incorporate a new crossover to improve the search ability. The next section describes our proposed model in detail.

3. PARALLEL MULTI-OBJECTIVE GENETIC ALGORITHM SUPPORTING HETEROGENEOUS COMPUTATIONAL RESOURCES

3.1 Basic Model

This section presents an explanation of the proposed parallel model of multi-objective GA. The proposed model extends the master-slave model, *i.e.*, the master process sends two individuals in the current population to each slave process. After each slave process receives two individuals, cross-over is performed several times and the number of offspring generated changes adapting to the performance of the calculation resource. There are two problems that should be examined in this case:

1. How are the two individuals transmitted by the master process to each slave process determined ? at random or according to a certain rule?
2. Does the search ability improve by increasing the number of offspring generated by each slave process?

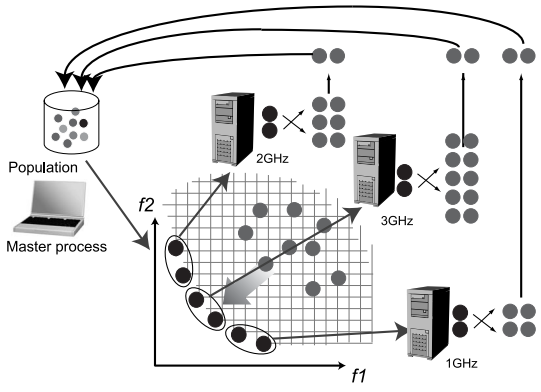

Figure 1: Basic model of the proposed parallel EMO

The basic model of the proposed parallel GA is illustrated in Figure.1. As the proposed method is based on NSGA-I I [7], the algorithm has the same flow. In this paper, the multi-objective GA is based on NSGA-II. However, other powerful GAs, such as SPEA 2 [8], can also be utilized. The proposed method differs from Multi-Objective GAs, such as NSGA-II, in two respects: neighborhood crossover and the number of generated offspring after crossover. In the neighborhood crossover, two individuals that are close each other are selected as parents. These two individuals are sent to the calculation resource. Then offspring are generated in each calculation resource. In this step, the number of offspring generated changes with the performance of the calculation resources. In this step, many offspring are generated on the high performance calculation resource and a smaller number of offspring are generated on the low performance calculation resource. Then, the two best offspring are chosen and returned to the master process. With this mechanism, a high degree of parallelization can be achieved and the generation continues synchronously.

3.2 Neighborhood Crossover

Effective crossover often cannot be performed in typical multi-objective genetic algorithms, as the search directions of each parent individual are different from each other. Therefore, we proposed neighborhood crossover, which generates offspring from two parent individuals neighboring each other in the objective space [9, 10]. By crossing over individuals that are close to each other, offspring can be generated near the parent individuals. Therefore, the search progresses maintaining diversity.

Neighborhood crossover is performed as follows:

1. From the best individual for one of the function values, the population is sorted in close order in the objective space.
2. Neighborhood shuffle, which changes individuals randomly in some range of population size, is performed for the sorted population to prevent crossing over repeatedly between the same pair of individuals.

3. Crossover is performed between two individuals located side by side.

In neighborhood crossover, the neighborhood shuffle operator is the key to effectiveness. We will discuss the effectiveness of neighborhood crossover in detail in section 4.

3.3 Increasing the Number of Offspring Generated

In the proposed model, the number of offspring generated is changed with the performance of the calculation resources. In conventional GA, two offspring are usually generated after crossover. We increase this number and select the best two offspring according to the following procedure:

1. Neighborhood crossover is performed several times depending on the performance of each calculation resource, and the offspring population C is formed.
2. The mutation operation is performed against C, and all offspring are then evaluated.
3. Non-Dominated Sort [7] is performed against C to rank all offspring.
4. The two best offspring that are rank1 and most excellent with regard to objective function values are returned to the parent population. If the number of rank 1 offspring is 1, the best offspring in rank 2 is then returned to the parent population.
5. Iterate from step 1 to step 4 against the parent population, and the archive is then updated.

In this algorithm, when the number of offspring generated by crossover increases, the number of evaluations per generation also increases.

Before the proposed parallel model is examined, it is necessary to investigate the effectiveness of neighborhood crossover and increasing the number of offspring in Multi-Objective GAs. First, the effectiveness of neighborhood crossover is discussed in section 4, and the effectiveness of increasing the number of offspring is discussed in section 5. Then, our proposed model is reviewed through the results of computational experiments on heterogeneous computational resources in section 6.

4. EFFECTIVENESS OF NEIGHBORHOOD CROSSOVER

As mentioned in the previous section, neighborhood shuffle is a very important operation in neighborhood crossover. If neighborhood shuffle is not conducted, crossover will be conducted with the same pair in every generation, and thus it will be impossible to escape after falling into a localized solution. Therefore, it is important to perform neighborhood shuffle with a moderate width. The width of neighborhood shuffle in which this neighborhood shuffle is conducted is decided depending on the ratio of the neighborhood shuffle width (R_{nsw}). R_{nsw} is a real number between 0 and 1.0, and the size of the width of neighborhood shuffle is represented as the ratio to the size of the population. For example, R_{nsw} 0.1 means that the neighborhood shuffle is conducted with a width that is 10% of the population. The proximity of the individuals changes depending on the size of R_{nsw}, and the proximity increases with reduction in this size, but this also increases the possibility that crossover will be conducted repeatedly with the same pair. This section describes the influence of changes in width of neighborhood shuffle on solution search ability examined through a numerical experiment, to investigate the effectiveness of neighborhood crossover.

4.1 Test Problems

In this study, NSGA-II using neighborhood crossover was applied to several types of test function. Due to the limitations of this paper, the discussions are described for the following two test functions that have wide Pareto-optimal front: KUR [11] and multi-objective 0/1 knapsack problem [8, 12] with two knapsacks and 750 items.

KUR is a problem with an interaction between two continuous variables in $f_1(x)$ and a multi-convex in $f_2(x)$. In this experiment, there were 100 design variables and it was very difficult to find the solutions.

While multi-objective 0/1 knapsack problems with two knapsacks and 750 items in Zitzler are easy to setup, the problem itself is very difficult and it is difficult to find the solution.

4.2 Performance Measures

To evaluate the derived Pareto-optimal solutions, two factors should be measured: accuracy and diversion. The derived Pareto-optimal solutions should be close to the real Pareto-optimal solutions. At the same time, the derived solutions should not concentrate on a certain point. The derived solutions should also be scattered over a wide area. For this purpose, various performance measures have been proposed to evaluate non-dominated solution sets. In this paper, we use the following performance measures to compare a number of solution sets simultaneously:

1. Cover Rate
2. Spread [13]
3. Hypervolume [14]
4. Ratio of Non-dominated Individuals: RNI [15]

The Cover Rate is a method of evaluating whether the solution set is distributed uniformly in the objective space and to calculate the rate of number k_i of the small domain when the domain of Pareto-optimal solutions is divided into K parts. The Cover Rate calculated for the solution set in N objective functions is as follows. The closer to 1.0, it is estimated that the solution can be found to all domains. In this experiment, we set the number of divisions K to the population size.

$$\text{Cover Rate} = \frac{1}{N} \sum_{i=1}^{N} \frac{k_i}{K}$$

Table 1: Parameters

Problem	KUR	KP750-2
Population Size	100	250
Number of Dimensions	100	
Chromosome Length	2000	750
Crossover Probability	1.0	
Mutation Probability	1/Chromosome Length	
Max Generation	250	2000

The Spread measure is a method of evaluating whether the solution set is obtained widely and can be calculated for the solution set as follows:

$$\text{Spread} = \sum_{i=1}^{N} [\max f_i(x) - \min f_i(x)]$$

The Hypervolume calculates the size of the dominated space by the obtained solutions in the objective space.

RNI is a method for evaluation by comparing the dominance of two populations obtained by two different algorithms. In RNI, the populations obtained from the two algorithms, S_1 and S_2, are combined to make a union set, S_U. The set of non-dominated individuals S_P is obtained from S_U. The number of individuals contained in S_P from each algorithm is used to obtain the ratio, and the value is used as the result of the evaluation. When the value is closer to the maximum of 100%, the algorithm has produced a better population.

4.3 Discussion through Numerical Experimentation

In this section, the effects of neighborhood crossover are discussed through numerical examples. In the conventional NSGA-II, the binary tournament selection is used as the mating selection method. In the proposed method, copy selection is used for mating selection as many individuals that are as different from each other as possible are needed in neighborhood crossover. The parameters used in this experiment are shown in Table 1.

In each target problem, various R_{nsw} are examined. In KUR, R_{nsw} with values of 0.0, 0.05, 0.1, 0.2, 0.25, 0.5, and 1.0 are examined. R_{nsw}0.0 means that neighborhood shuffle is not executed after the sort, and as R_{nsw}1.0 executes neighborhood shuffle in width of the size of the population after the sort, it will be the same as the original NSGA-I I which changes the selection method only. In addition, in KP750-2, R_{nsw} with values 0.0, 0.02, 0.04, 0.1, 0.2, 0.5, and 1.0 are examined.

The results for KUR and KP750-2 are shown in Figures. 2 and 3, respectively. The results of original NSGA-II are also shown for comparison. The results are shown as averages of the 30 runs. Figures. 2(a) and 3(a) show the Cover Rate, 2(b) and 3(b) show Spread, 2(c)and 3(c) show Hypervolume, and 2(d) and 3(d) show RNI compared to the original NSGA-II. Figures. 2 and 3 indicate that the obtained solution sets with neighborhood crossover are wide and have excellent diversity, especially when R_{nsw} is around 0.1 or 0.2.

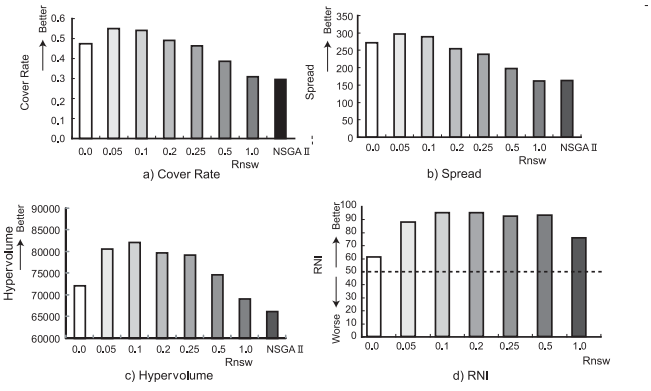

Figure 2: Results of Cover Rate, Spread, Hypervolume and RNI when R_{nsw} is changed in KUR problem

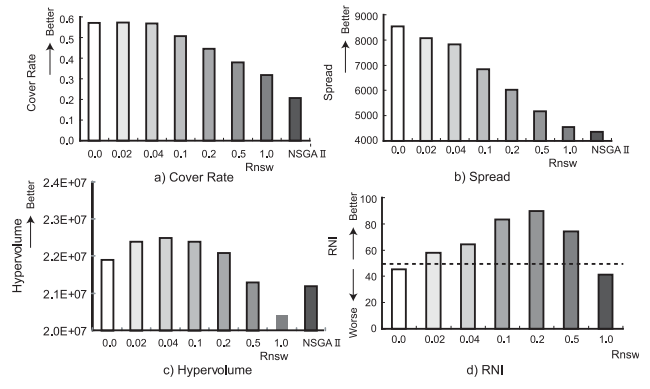

Figure 3: Results of Cover Rate, Spread, Hypervolume and RNI when R_{nsw} is changed in Knapsack problem

When $R_{nsw} = 0.0$ (*i.e.*, when neighborhood shuffle is not conducted), the search was believed to be influenced because the frequency of crossover in the same pair increased. These results confirmed that the best Pareto-optimal solutions are obtained when the neighborhood shuffle is conducted with the some width of neighborhood shuffle.

5. EFFECTIVENESS OF INCREASING THE NUMBER OF OFFSPRING GENERATED

In this section, we clarify the effects of the number of offspring through numerical experiments. The target functions and parameters used are the same as in the previous section. As the effect of the number of offspring is of interest, the same number of offspring were generated from the two parent individuals.

5.1 Numerical Experiments

To clarify the effects of increasing the number of offspring, we performed numerical experiments with the number of offspring generated by neighborhood crossover set to 2, 4,

6, 8, 10, and 20. The number of evaluation calculations was fixed to 25,000 in KUR and 500,000 in KP750-2.

The results for KUR are shown as averages of the 30 runs in Figure. 4. As shown in Figure. 4(a), the diversity was lost with increasing number of offspring. Figures. 4(c) and (d) also show that the performance was inferior to the original NSGA-Ⅱ as the number of offspring increased. If the number of offspring increases, the number of evaluations per generation must also increase. Thus, with this algorithm, both the number of generations and the performance were reduced as compared with the original NSGA-Ⅱ with the same number of evaluations. On the other hand, Figure. 4(b) indicates that Spread was improved with increases in the number of offspring.

The results for KP750-2 are also shown as averages of the 30 runs in Figure. 5. As shown in this figure, the Cover Rate and Hypervolume were inferior when the number of offspring was set to 20, and the RNI was inferior with increases in the number of offspring. On the other hand, the Spread improved with increases in the number of offspring. These results indicate that increasing the number of offspring causes convergence to be slow because of the decrease in number of generations, but the solution set obtained is wide in the objective space.

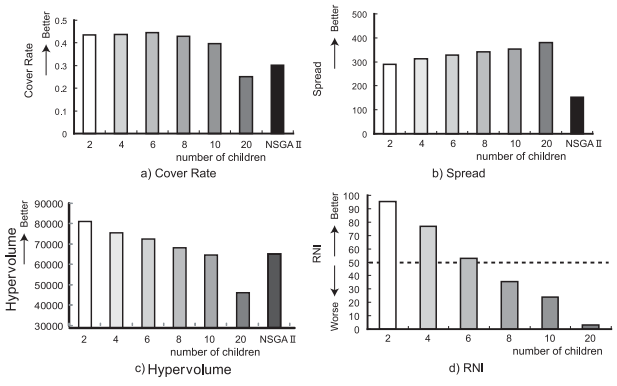

Figure 4: Results of Cover Rate, Spread, Hypervolume, and RNI with the number of evaluations fixed in KUR proble

Then, we compared the results with those when the number of generations was fixed. As explained above, in the proposed model, the number of offspring generated changes with the performance of the computational resources. Therefore, it is important to discuss not only fixed total evaluation number but also fixed total number of generations. The effects of increasing the number of offspring can be seen by fixing the number of generations. If the performance is improved with increases in the number of offspring, it would make sense to increase the number of offspring according to the performance of the calculation resources available.

The results for KUR with the number of generations fixed to 50 are shown in Figure. 6. The results for KP750-2 with the number of generations fixed to 200 are also shown in Figure. 7.

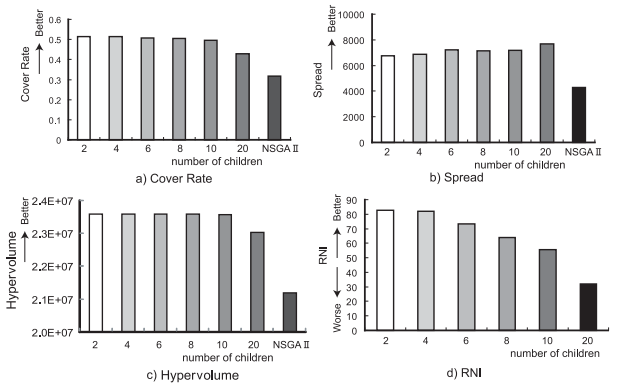

Figure 5: Results of Cover Rate, Spread, Hypervolume, and RNI with the number of evaluations fixed in KP750-2 problem

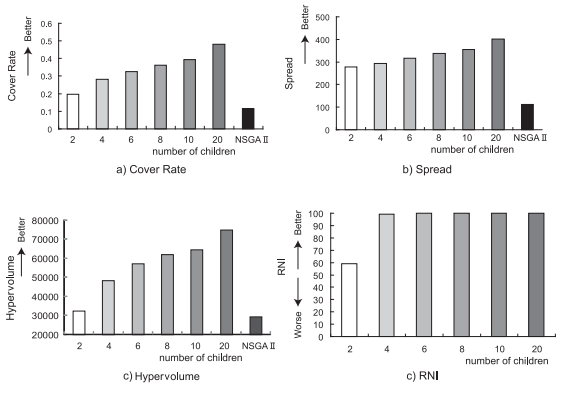

Figure 6: Results of Cover Rate, Spread, Hypervolume, and RNI with the number of generations fixed in KUR problem

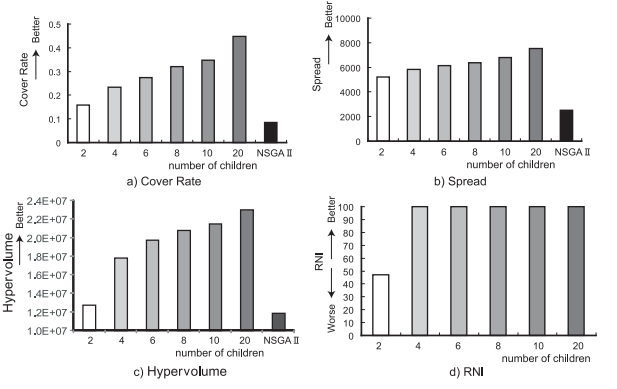

Figure 7: Results of Cover Rate, Spread, Hypervolume, and RNI with the number of generations fixed in Knapsack problem

Figures. 6 and 7 show that the diversity improves and the obtained solution sets become wider as the number of offspring increases. The distribution graph in Figure.8 shows the results of collecting all solutions for 30 runs with the number of offspring set to 2, 10, or 20 for KUR. Figure. 9 also shows the distribution graph with the number of offspring set to 2, 6, or 10 for KP750-2. From Figures. 8 and 9, we can see the effects of increasing the number of offspring.

5.2 Discussion

Although the performance improves with increases in the number of offspring generated by the crossover operation, the number of evaluations per generation also increases. Therefore, in comparison with the original multi-objective genetic algorithm for the same number of evaluations, it becomes difficult to achieve a sufficient number of generations and yields inferior results. On the other hand, when we performed the comparison with a fixed number of generations, it was possible to obtain a wider variety of offspring in the objective space as the number of offspring increased.

From the observations described above, it is expected that a Pareto-optimal solution set with a high degree of accuracy can be obtained by increasing the number of offspring according to the performance capabilities of the available calculation resources. That is, when two individuals transmit to each calculation resource at the same time, those resources with excellent performance would generate large numbers of offspring, while those with inferior performance would generate few offspring, and the execution time per generation can be united.

6. COMPUTATIONAL EXPERIMENTS ON HETEROGENEOUS COMPUTATIONAL RESOURCES

In this section, we clarify the effects of the proposed parallel model through computational experiments on heterogeneous computational resources.

6.1 Experimental environment and Procedure

In this experiment, the validity of the proposed parallel model was verified using one master process and a total of 50 slave processes using a 4-PC Cluster comprised of PCs that differed in performance. Calculation resources are shown in Table 2. We used the Grid RPC Ninf-G (version 2.4) [16] to submit jobs to each PC Cluster, and Open-PBS (version 1.2) was used for scheduling jobs in each PC cluster. Ninf-G is a reference implementation of the Grid RPC system using the Globus Toolkit [17].

The flow of execution is shown in Figure. 10. First, the master process generates an initial population and performs Neighborhood Sort, which reorders in close order in objective space and also perform neighborhood shuffle. Then, the master process submits two adjacent individuals to master nodes of each PC Cluster using Ninf-G. At this time, the data transmitted are the chromosome information of two individuals. After the master nodes of each PC Cluster receive these data, they begin scheduling the jobs and distribute them to slave processes. Each slave process repeats the operations of crossover, mutation, and evaluation over a given time, and perform non-dominated sort to choose the best two offspring that should be returned to the master process. The data transmitted from slave processes to the master process are chromosome information and objective function values of the best two offspring. These procedures are carried out on all slave processes with synchronous communications in one generation.

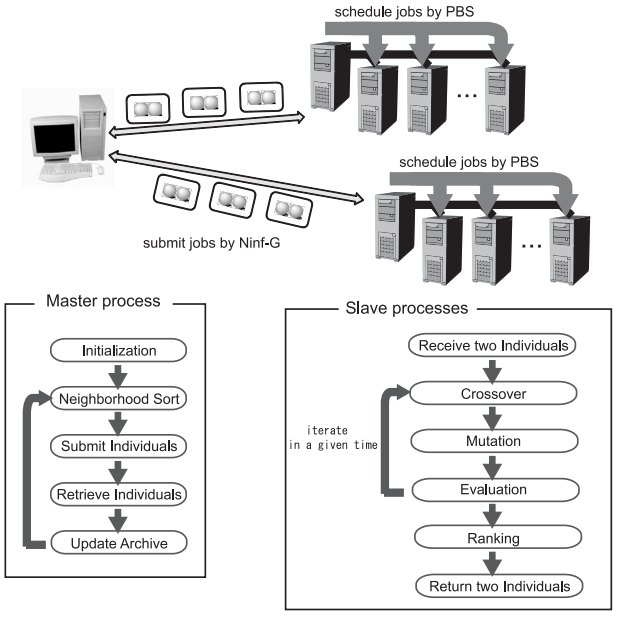

Figure 10: Execution Flow of Proposed model on heterogeneous computational resources

In this experiment, each calculation resource increased the number of offspring adapted for performance of the calculation resources by repeated crossover, mutation, and evaluation during a given time. Thereby, all the calculation resources can terminate processing almost simultaneously after a given time, and the delay by the calculation resource with low performance can be prevented. It is necessary to set up a fixed time based on the calculation load of the object problem on the performance of calculation resources.

The target function is KUR the calculation load of which is increased by executing useless calculations to assume a real problem. The evaluation times of one individual by the calculation resources of each PC cluster for this problem were 5.82 s on PC Cluster A, 8.62 s on B, 10.17 s on C, and 17.06 s on D. Then, we set up a fixed time of 1 min so that the calculation resources of PC Cluster D, which has the poorest performance, could generate at least 2 offspring.

We compared our proposed parallel model with the original NSGA-II algorithm, which is parallelized with the original master slave model under the same environment and conditions, and set 2 h as the termination condition. The parameters used in this experiment were the same as in the previous section: population size, 100; number of dimensions, 100; R_{nsw} in our proposed algorithm was set to 0.1.

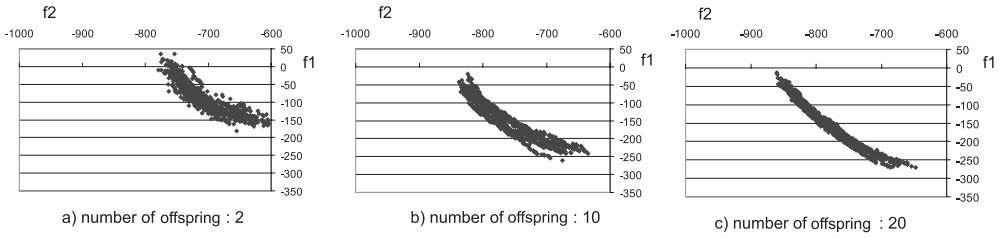

Figure 8: Obtained pareto-optimal solutions with the number of offspring set to 2, 10, or 20 in KUR problem

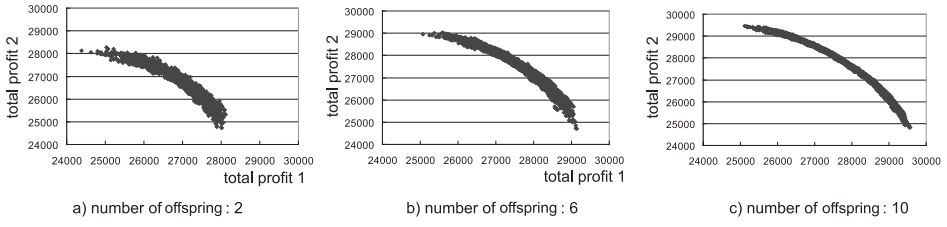

Figure 9: Obtained pareto-optimal solutions with the number of offspring set to 2, 6, or 10 in Knapsack problem

6.2 Experimental Results

The average results over three runs for the KUR test problem are summarized in Figure. 11. As shown in this figure, the effectiveness of the proposed model increased sharply, especially in diversity and spread of Pareto-optimal solutions. That is, the proposed parallel model was confirmed to be effective on heterogeneous computational resources. Figure. 12 shows the distribution graph of all three runs obtained by the proposed model and the original master-slave model. Figure. 12 shows that the solution sets obtained using the proposed model are good in terms of both diversity and spread.

6.3 Discussion

In the proposed model, all calculation resources can terminate processing almost simultaneously after a given time so that the idle time of all calculation resources can be maintained to the minimum. It is also possible to reduce the overhead time, such as communication time or jobs scheduling, by increasing the process in remote slave processes. Table 3 lists average CPU usage rates of a process in each PC cluster, and Table 4 lists the total overhead time. Table 3 indicates that the idle time becomes longer on high performance calculation resources, especially PC cluster A, in the original master-slave model. However, all processes have uniform loads in the proposed model. The data shown in Table 4 also confirmed that the influence of overhead time is reduced in the proposed parallel model.

Finally, Figure. 12 shows the search history of the proposed model and the original master-slave model that plots solution sets at times of 30, 60, and 120 min of progress. In comparison with the original NSGA-II search process, wide-ranging non-dominated solutions with great diversity were obtained from the early stages of the search. These observations indicate that it is possible to conduct a search while maintaining the diversity of the population, and to obtain a wide range of Pareto-optimal solutions using the proposed model.

Table 3: Average CPU usage rate of a process in each PC cluster

	A	B	C	D
Proposed parallel model	88%	91%	95%	91%
Original master-slave model	18%	44%	53%	89%

Table 4: Overhead time in the proposed model and the original master-slave model

	Time
Proposed model	4m5s
Original master-slave	4m38s

7. CONCLUSIONS

In this paper, we proposed a new parallel model of EMO supporting a heterogeneous environment and examined the accuracy of this model through computational experiments. We combined our neighborhood crossover with a multi-objective genetic algorithm. We also considered increasing the number of offspring dynamically according to the performance of the available calculation resources. The results of computational experiments indicated that neighborhood crossover performs well and it was possible to obtain a wider variety of individuals in the objective space as the number of offspring increased. We also investigated the validity of the proposed parallel model on heterogeneous calculation

Table 2: Calculation Resources

	number of CPU	CPU	Memory	OS
Client	1	Athlon64 3200+	1GB	Fedora Core 4
PC Cluster A	10	Pentium4 2.8GHz	1GB	Debian 3.1
PC Cluster B	15	Xeon 2.4GHz	1GB	Debian 3.1
PC Cluster C	15	PentiumIII 1GHz	512MB	Debian 3.1
PC Cluster D	10	PentiumIII 600MHz	256MB	Debian 3.1

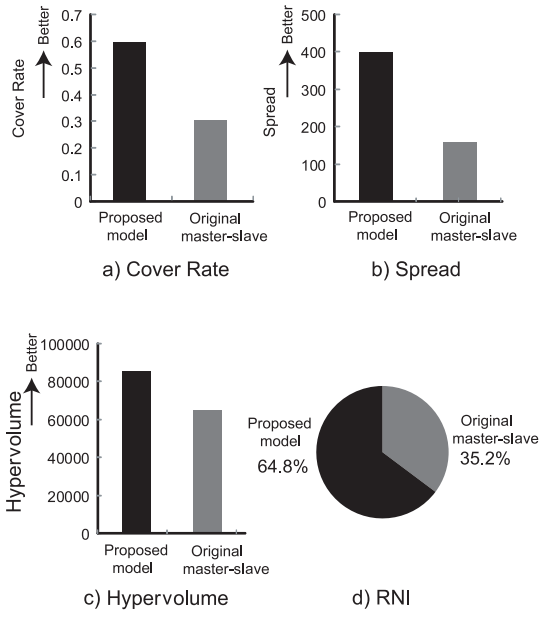

Figure 11: Results of Cover Rate, Spread, Hypervolume, and RNI on proposed parallel model and original master-slave model in KUR problem

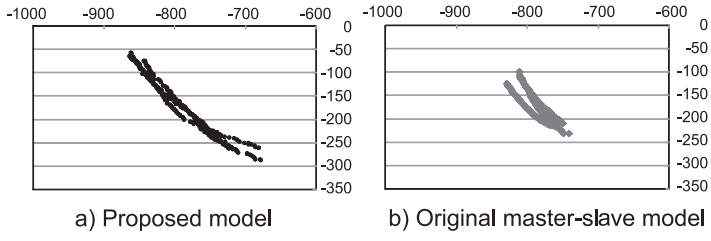

Figure 12: Pareto-optimal solutions obtained with the proposed model and the original master-slave model in KUR problem

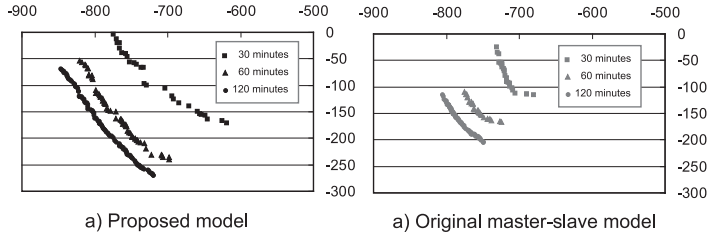

Figure 13: Search history of the proposed model and the original master-slave model

resources. Computational experiments indicated that the proposed model has high search ability, and was able to utilize the maximum performance of all calculation resources and reduce the overhead time.

8. REFERENCES

[1] Kalyanmoy Deb, Pawan Zope, and Abhishek Jain. Distributed computing of pareto-optimal solutions with evolutionary algorithms. In *EMO*, pages 534–549, 2003.

[2] David A. van Veldhuizen, Jesse B. Zydallis, and Gary B. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 7(2):144–173, 2003.

[3] F. Streichert, H. Ulmer, and A. Zell. Parallelization of multi-objective evolutionary algorithms using clustering algorithms. In Carlos A. Coello Coello, Arturo Hernandez Aguirre, and Eckart Ziztler, editors, *Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 of *LNCS*, pages 92–107, Guanajuato, Mexico, 9-11 March 2005.

[4] F. de Toro Negro, J. Ortega, E. Ros, S. Mota, B. Paechter, and J. M. Martín. Psfga: parallel processing and evolutionary computation for multiobjective optimisation. *Parallel Comput.*, 30(5-6):721–739, 2004.

[5] Gary B. Lamont Carlos A. Coello and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[6] Jürgen Branke, Hartmut Schmeck, Kalyanmoy Deb, and M. Reddy. Parallelizing multi-objective evolutionary algorithms: cone separation. In *Congress on Evolutionary Computation*, volume 2, pages 1952–1957. IEEE, JUN 2004.

[7] A. Pratab K. Deb, S. Agrawal and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[8] M. Laumanns E. Zitzler and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[9] T. Hiroyasu S. Watanabe and M. Miki. Neighborhood cultivation genetic algorithm for multi-objective optimization problems. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL-2002)*, pages 198–202, 2002.

[10] T. Hiroyasu M. Kim and M. Miki. Spea2+: Improving the

performance of the strength pareto evolutionary algorithm2. *Parallel Problem Solving from Nature - PPSN VIII*, pages 742–751, 2004.

[11] F. Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[12] E. Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.

[13] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 1998.

[14] K. Deb E. Zitzler and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[15] T. H. Lee K. C. Tan and E. F. Khor. Increamenting Multi-objective Evolutionary Algorithms: Performance Studies and Comparisons. *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 111–125, 2001.

[16] "Yoshio Tanaka, Hidemoto Nakada, Satoshi Sekiguchi, Toyotaro Suzumura, and Satoshi Matsuoka". "Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing". In *Journal of Grid Computing, Vol. 1, No. 1, pp. 41-51, Kluwer Academic Publishers, June, 2003.*, 2003.

[17] The Globus Alliance. . `http://www.globus.org/`.