Discussion on Searching Capability of Distributed Genetic Algorithm on The Grid

Yusuke Tanimura
Graduate School of Doshisha Univ.
1-3 Tatara Miyakodani
Kyotanabe, Kyoto 610-0331
tanisuke@mikilab.doshisha.ac.jp

Tomoyuki Hiroyasu
Dept. of Engineering, Doshisha Univ.
1-3 Tatara Miyakodani
Kyotanabe, Kyoto 610-0331
tomo@is.doshisha.ac.jp

Mitsunori Miki
Dept. of Engineering, Doshisha Univ.
1-3 Tatara Miyakodani
Kyotanabe, Kyoto 610-0331
mmiki@mail.doshisha.ac.jp

Abstract- The computational Grid has become popular recently. Since the Grid has the tremendous power, it is expected that the numerical optimization method like Genetic Algorithms (GA) performs well on the Grid. In the former works, only the simple model of GA is applied on the Grid. In this paper, when the distributed GA (DGA) is executed on the Grid, the considerable issues and problems are discussed for the Scalability, Dynamic Changes and Heterogeneity. Through the numerical experiments, it is found that the DGA model has the following features on the grid; DGA has the scalability for searching the solutions with respect to the number of the resources and the results of the DGA are not influenced very much by the dynamic reduction of the number of resources. It is also addressed the affect of the asynchronous migration. As a result, the guideline how to implement the DGA on the Grid is described.

1 Introduction

In these days, the computational Grid is being paid attention as the environment for huge calculation of science and technology[foster98]. The Grid is an infrastructure where users can use several types of resources that are connected by the Net freely like the Power Grid. Since there is a possibility for users who can use tremendous computational resources, many application users have been interested in the Grid recently. However, the Grid has several characteristics; the resources are changing dynamically, the network latency is very huge, and so on. Because of these characteristics, not all of applications are suitable for the computational Grid. Genetic Algorithm (GA) is one of the applications that are suitable for the Grid. Because GA is necessary to remarkably more computation power than the conventional optimization method, the use of the Grid is a key theme for GA. In this paper, the issues and probabilities of effectiveness of the distributed GA (DGA) that is performed on the Grid are discussed.

Recently, several types of Grid testbeds have been constructed and being operated. On these testbeds, GA was implemented and studied as a test program for the Grid-oriented execution model[hcw00shao]. Those previous researches, however, are the parameter-sweep implementation model where several parameters have been already prepared or where the parameters are being searched[sc00casanova]. There are other researches that are concerned with the master-slave implementation model where the calculation of the value of fitness function is submitted to other servers[hcw00shao]. Since the master-slave model is not a computational model but an implementation model in many cases, the discussions are not on the GA searching capability but on the system evaluations. Therefore, more discussions on the searching capability of GAs are needed.

The goal of this research is to make clear the possibility and issue to utilize DGA on the Grid with consideration of the search performance. In this paper, the basic DGA model is extended to have the dynamic reduction of the sub population and the asynchronous migration. The discussions are performed on the influence of searching scalability with along to the number of resources, the dynamic reduction of the sub population, and the asynchronous migration.

2 Distributed Genetic Algorithm and the Grid Model

2.1 Distributed Genetic Algorithm

GA is a numerical optimization method that imitates the biological evolution and selection[goldberg89]. It performs multi-point and probabilistic search. The GA can find the optimum solution even in discreet optimization problems where the conventional methods that need the differential value of the searching point cannot performed. The GA, however, needs large calculation costs because it contains a lot of iterations. As one of the solutions for this problem, there are several investigations of parallel GAs[comp99alba, cantu-paz00, eaecs99tomassini].

In this paper, the DGA model, one of the representative parallel GAs known as the island model, is focused. In the DGA model, the biological specialization is applied and the superior performance is provided. It is also suitable to parallel execution because of the comparatively coarse-grained communication. In the DGA model, the total pop-

ulation is divided into several sub-populations and the independent search is performed in each sub population. This sub population is often called an island. After several generations, some individuals (searching points) are exchanged among islands. This operation is called migration. Besides crossover rate and mutation rate, there are other parameters in DGA: the interval of migration, the number of migrates, and the migration topology. These parameters are defined by the user.

There are other types of parallel GAs; the typical models are the Master-slave model and the Cellular model. The Master-slave model is an implementation model rather than a computational model. Therefore, the discussion parts exists in system, not search algorithm. This model can be implemented using the well-known Grid system like the NetSolve[sahpc97casanova]. Since operations of the Master-slave model of GA are similar to the other Master-slave models, there are some relative studies[grid02nakada, anl00goux]. It is also possible to implement the DGA model as the Master-slave model. However, in this paper, the DGA is implemented as a model where each CPU or node has its own island.

The Cellular model is an implementation as a parallel model exploiting the block division or cyclic division like the case of the matrix operation. Since the fine-grained communication is compulsory in this model, the model should be changed a lot when this model is used on the Grid.

2.2 Characteristics of The Computational Grid

On the computational Grid, many computational resources are connected by the Net. Users can use these huge computational resources. There, however, are several characteristics that has been explained before. The important characteristics of the Grid for executing DGA model are summarized as follows;

- Huge Computational Resources
  The large-scale GA can be performed on the Grid, because the available resources may be infinite. Nevertheless, the conventional GA is created in the framework of limited resources. There is scalability issue for a new model or extended model of GA. The simplest implementation for GA to use these huge resources is an algorithm where the number of computational resources becomes bigger, the population size or the number of islands makes bigger. However, it is not guaranteed in GA that bigger population leads the better solutions. Therefore, the discussion for the effective usage of computational resources is necessary. The big issue is whether the reasonable precision can be got or not on spending resources.
- Dynamic Change (Load Average, Drawback)
  The Grid is consisted of several remote resources on the wide-area network and built for some specified purposes by some communities. There are multiple users on the Grid and most resources are shared. Because of these situations, the computational environment can be changed as time is going. The search model should accept this dynamic environment. In addition, the frequency of the drawback is high. When the drawback happens, the search model should continue the present calculation without recalculation from the initial.
- Heterogeneity
  The resources on the Grid are provided by several different organizations and they are heterogeneous. The variation of the architecture and compiler should be solved by the Grid middleware without giving constraints to developers or users. The heterogeneity of the performance, however, cannot be solved by it. To execute in parallel, the search model should have the algorithms suitable to the performance heterogeneity. It involves an asynchronous communication model inevitably.
- Communication Topology
  In case of executing the large-scale DGA, the effective communication model to exchange the information of searching points is essential. Compared with the small-scale execution, the delay to transmit information, the higher overhead, and the dynamic change of throughput might exist. Although the stepping stone topology is regularly adopted in the DGA model, the new topology to solve the problems about slow communication is mandatory on the Grid.

2.3 DGA Model on the Grid

In the former section, the characteristics of the Grid are summarized. Because of these characteristics, we think the model of DGA on the Grid should have the following features;

- Variable Population:
  Total population should be variable with respect to the total calculation resources. In this paper, population size in each island is fixed and the number of islands is variable.
- Dynamic Population:
  It may happen that some calculation resources are disappeared or emerged during the search. Therefore, the population should be dynamic.
- Asynchronous Communication:
  Since the calculation power of each node of the Grid is different, asynchronous communication should be

performed. Otherwise, there is a huge over head. However, when asynchronous communication is performed, the evolutionary speed of each island may be different.

- Tree Topology Communication:
  Most of the cases, the Grid consists of clusters, gateways, and many nodes. Therefore, the topology of computer resources is a tree topology. Therefore, it is efficient that the communication topology is a tree. In this paper, this discussion is not performed but in the future work.

2.4 Discussion Points and Numerical Experiments

In this paper, the influences of the Grid characteristics to the results of DGA is discussed. The discussion points are summarized as follows;

- Searching scalability for the number of resources.
- Influence to the results by dynamic reduction of the number of islands.
- Influence to the results by asynchronous migration.

In this paper, these points are examined through the simulation where the DGA is applied to solve the test functions. The parameters of the basic DGA model are showed in Table 1. These parameters are basically determined from the results of the former study [ipsj02hiroyasu].

In the experiments, three objective functions which would be minimized are typically continuous test function. They are not appropriate on the Grid for their small calculation cost but they are applied in order to discuss the above issues. They are the Rastrigin function[pc91muhlenbein], the Rosenbrock function[um75dejong], and the Ridge function[rechenberg94]. The Rastrigin function has a lot of local optimum and it is not easy to find the global optimum by the conventional gradient method. The Rosenbrock function has a globally large peak. This function, however, is difficult to solve by GA because the design variable depends on each other and the crossover operation does not work well. The Ridge function has weak dependency for the design variable and a globally large peak.

The number of design variables is set to 30 in those problems. The well-known optimum is in that $F(x) = 0$ in all the problems. In all experiments except Chapter 5, the values of the results are derived by the average of 50 trials.

3 Scalability for Searching

The discussion is whether DGA has the searching scalability with respect to the population size or the number of islands. Since one island is allocated on one node or CPU in this paper, the increase of resources means the increase of the islands. There is a possibility that the evolution of GA becomes slow, because increasing the searching points more than the specified number makes extraordinary solution diversity and the search slow. In that case, the large-scale GA just wastes resources and it is never efficient.

Table 1: Used Parameters of DGA

# Individuals/Island	12
Chromosome length (# Design variables)	300 (30)
Crossover method (Rate)	1pt. Crossover (1.0)
Mutation method (Rate)	Bit complement (1 / Chromosome length)
Selection method	Roulette
Migration topology	Ring generated randomly each time
Migration interval	5 generations
Migration rate	0.15

3.1 Implementation and Experiment

The ga2k (Ver.1.4)[ga2kspec3], one of the implementations of DGAs is utilized and extended for the experiments in this paper. There is an experiment to investigate the search performance from 256 islands to 5020 islands with the fixed population size of 12 in each island.

3.2 Results

The number of individuals in one island is fixed to 12. Then the number of islands is made increase from 256 to 5020 in this experiment. The search performance is showed at Figure 1. In case of the Rastrigin, the optimum was always to be derived, but not in other cases. Therefore, in case of the Rastrigin, the vertical axis shows the generation that found the optimum. In other problems, the vertical axis shows the best solution value that found within the 2000th generation.

In the results of the Rastrigin and Rosenbrock, the trial with more searching points can find better solutions. The similar tendency can be found in the Ridge function, but decreasing has saturation in the trials with large number of islands. From these results, it is found that the increase of the computational resources does not guarantee the linear scale up for the execution time to find the optimum solution. The bigger computational resources, however, lead to the higher possibility to find the optimum solution.

4 Influence by Dynamic Reduction Model

The number of the searching points may be changeable after launching GA on the Grid. While other users are requiring more resources, the scale of GA can be keeping small. Otherwise, the scale can be keeping large, because of the

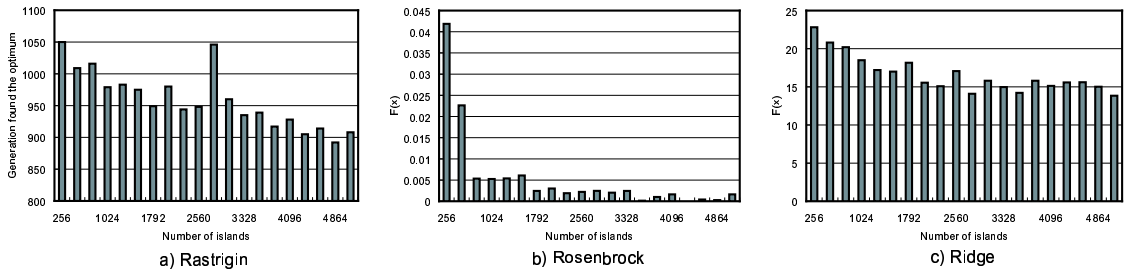

Figure 1: Scalability of DGA model

results that are derived in the former section. At the same time, since the latency sometimes becomes very large on the Grid, some nodes looks died. The possibility of the sudden death of the nodes is much higher than local calculation resources. Therefore, the affect of the reduction of the resources should be analized.

The used implementation is the same as the former section. The function to reduce the scale dynamically is implemented based on the ga2k. This function deletes islands randomly with the specified number in specified generation. For the migration topology is generated at random every migrating generation, the new topology shapes with the rest islands after reduction. The experiment to inspect the search performance is operated using the DGA model added the dynamic reduction. The main issues of this experiment are the reduction timing and the amounts of reduction.

4.1 Results

4.1.1 Reduction Timing

First of all, the transition of the fitness value variance for the normal DGA is investigated. Usually, in the GA search, there are three phases in the transition of variance[seal02sano]. The example of the Rastrigin function that is minimized by normal DGA is shown in Figure2. Phase I is the phase of the rapid convergence. Phase II is the slow convergence phase. Phase III is the steady state phase. In this experiment, the reduction is performed in each phase. For example, in case of Figure 3, a), the phase I is corresponded to 0 – 100th generation. In this experiment, the reduction is performed in 50th generation. The result of the influence for the search performance is showed in Figure 3, 4, and 5.

In the discussion of the reduction timing, 24 islands are reduced from the initial number of islands 64. In the graph b), the reduction in Phase I influences to the performance badly on all the problems. In the reduction in Phase II, the Rastrigin is suffered from unpleasant influence and the Ridge suffers from good influence. In Phase III, only the Rosenbrock suffers from bad influence and other problems

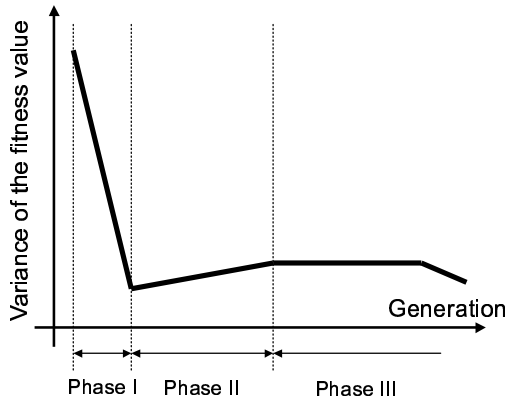

Figure 2: Search Phase

have no influence or a little good influence. From these results, there is higher risk for the reduction in Phase I and II. The reduction in Phase III has lower risk.

4.1.2 Amounts of Reduction

The experiment for the amounts of reduction is performed in this section. The reduction is performed in Phase I, which made the search performance the most influent. The amounts of reduction are 16, 24 or 32 islands from initial 64 islands. The c) in the graph shows that small amounts of reduction does not affect the solution but the huge amounts of reduction caused the worse search performance. This result suggests that users need not care for the small amounts of the reduction of the computational resources.

5 Asynchronous Migration and Differency of Evolution Speed

In the simple execution model of the DGA model, one island is assigned to one computation resource. One specified topology (mostly, the ring topology) is generated in the every migrating operation. This is assumed that all available computation resources have homogeneous performance and then all islands should be synchronized every migration. In the heterogeneous environment, there is happened some

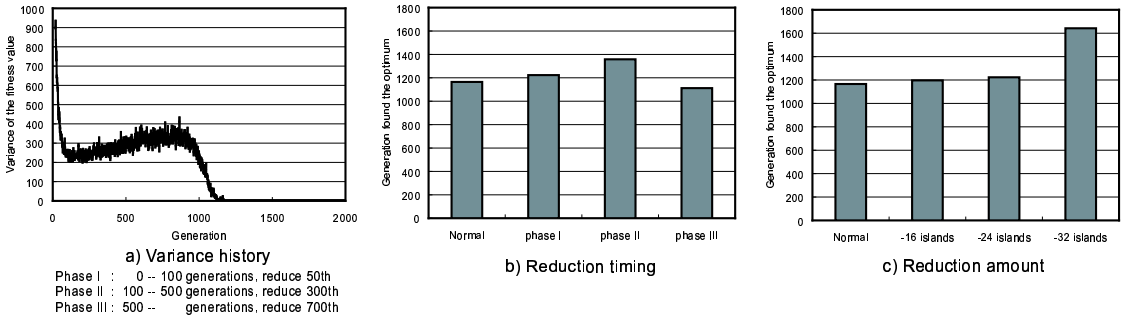

Figure 3: Reduction Affection (Rastrigin)

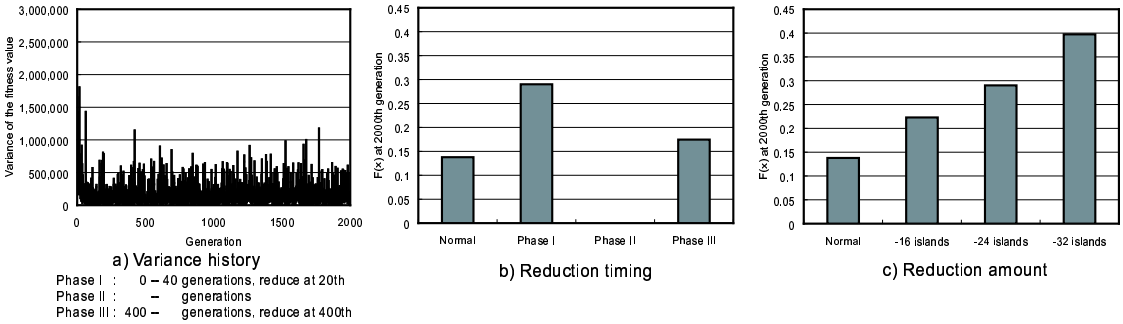

Figure 4: Reduction Affection (Rosenbrock)

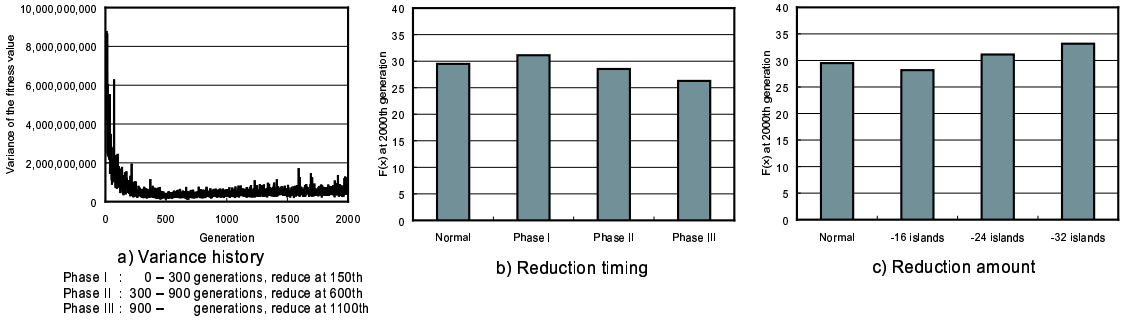

Figure 5: Reduction Affection (Ridge)

overhead for those migrations. To solve this problem, the asynchronous migration model is developed in this paper. The asynchronous migration creates another new problem that each island has different evolution speed. The effect to the search performance for it is examined in this chapter.

5.1 Implementation Using EVOLVE/G System

The parallel execution program of the DGA model is developed using the EVOLVE/G system[pdcs02tanimura] and based on the ga2k program. The EVOLVE/G system works with one Agent unit and several Worker units. These units communicate with each other and exchange application data. In our implementation, one Worker performs the one-island GA and Agent operates the migration among the islands on Workers. Agent checks Worker periodically and, 1) get the emigrants from Worker, 2) create the topology and apportion the migrates to islands and 3) put the immigrants to Worker. This series of operations is iterated at regular intervals.

The asynchronous migration is implemented as the topology is generated among Workers that Agent can have checked. It is showed in Figure 6. There are two asynchronous models in this paper. One of them is to move on the next generation after receiving the immigrants. The other is to move on the next generation without receiving the immigrants. Non-received immigrants might be combined to the island at the next migration. The former is called *semi-asynchronous migration*. The latter is called *full-asynchronous migration.* In the semi-asynchronous migration, the larger overhead may happen because Workers have to wait for a while to get the immigrants from Agent.

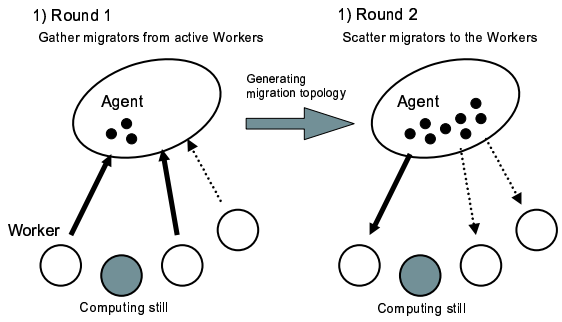

Figure 6: Implementation of DGA with Asynchronous Migration Using EVOLVE/G System

5.2 Results

The PC cluster system that consisted of 12 CPUs is used in this experiment. The performance of CPU is heterogeneous and their abilities are showed in Table 2. Each node is connected with 100 BASE Ethernet. The two models, Type A and Type B, are compared in this experiment. Type A is a model where asynchronous migration is performed without consideration of the performance heterogeneity. This situation causes the different evolution speed. In the heterogeneous computational environment, after certain execution time, many generations have been performed in some islands and few generations have been performed in other islands. We call this situation that evolution speed is different among the islands. Type B is a model where the number of individuals is arranged to match to the CPU performance. This operation means that the CPU that has the higher performance has the bigger population size. Because of the mechanism, the evolution speed becomes almost the same and the asynchronous migration does not happen so frequently. The number of individuals in Type A and B is also showed in Table 2. The total population is fixed and that is 144. Below, the results show only one trial but they are an example of multiple trials.

Firstly, the evolution speed of the DGA model with asynchronous migration is examined in Type A and B. Because the calculation cost of the test function is too small to evaluate these examinations, the 50,000 loops are performed in deriving one evaluation for taking time. Consequently, the difference of the evolution speed can be found in Figure 7 though the calculation and search cost is equal.

Secondly, the search performance is investigated in the different evolution speed. The result of the Rastrigin is shown in Figure 8. In this graph, the best $F(x)$ value at the time when Agent gathers the solution from Worker units cyclically is plotted. Type B could get a little better solution than Type A or similar with it in these results. The same results are shown in Figure 9 that the horizontal axis shows the generation. In this experiment, there is very little influence for the gap of the evolution speed.

Compared with the semi-asynchronous migration and the full-asynchronous migration, the former has more overhead to receive immigrants and its execution time is much slower. When the calculation cost for each individual is very large, that overhead might be hidden. The latter is better way to apply to the problem whose calculation cost for the individual is small. At the view point of the search performance, the full-asynchronous migration could find the solution well in both Type A and Type B from Figure 9. Since the semi-asynchronous migration usually has the synchronization in receiving immigrants, the migration is performed among islands that are the same evolution speed. On the other hand, in the full-asynchronous migration model, the immigrants are generated in the different generation. These are the difference between the full-asynchronous and the semi-asynchronous. These differences may affect the results.

The final experiment is for looking into details of migration between islands that have a different evolution speed. To perform this experiment, two models are prepared. In this experiment, the evolution speed is also defined by generation. The *Keep old* model is that the migration from the advanced island is not performed. The *Keep new* model is that the migration from the backward island is not performed. These two models are implemented with Type A at the semi-asynchronous migration. The normal model and they are compared in Figure 10. The normal model showed the best performance and the *Keep new* model also derived almost the same performance as it. On the other hand, the performance of the *Keep old* model was not good. In case of the DGA model that the evolution speed is different, the advanced island may get initial convergence. The immigrants from the backward island seem that they cannot participate in the new island. The immigrants from the advanced island might affect to the backward island's search process. They are possible to find another solution that is difficult to find in their source island. This phenomenon can be appeared in the final searching process. In Figure 10, the *Keep new* model was able to find the optimum earlier than the normal one.

6 Discussions

The experiments for Scalability, Dynamic Reduction and Asynchronous Migration on Heterogeneous Environment lead the following suggestions. When the computation resource is available sufficiently, the better solution can be obtained by using more resources. However, the accuracy for the solution is not increasing lineally even when the number of resources becomes bigger.

When the reduction of the island is happened, user can

Table 2: Heterogenious test environement

Host name	CPU type	# individuals (Type A)	# individuals (Type B)
Node 1 – 4	Pentium III, 850MHz	12	16
Node 5 – 8	Pentium III, 600MHz	12	11
Node 9 – 12	Pentium III, 500MHz	12	9

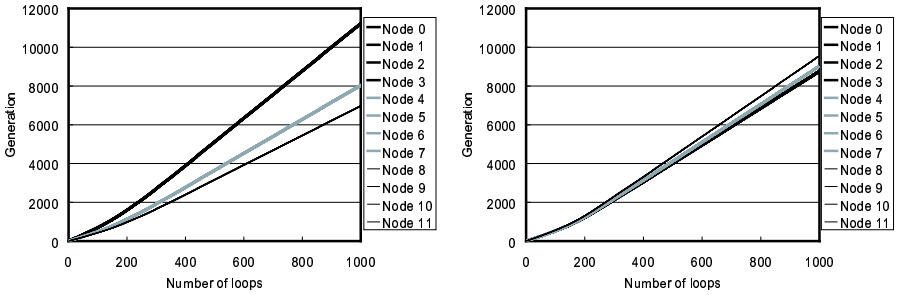

Figure 7: Evolution Speed

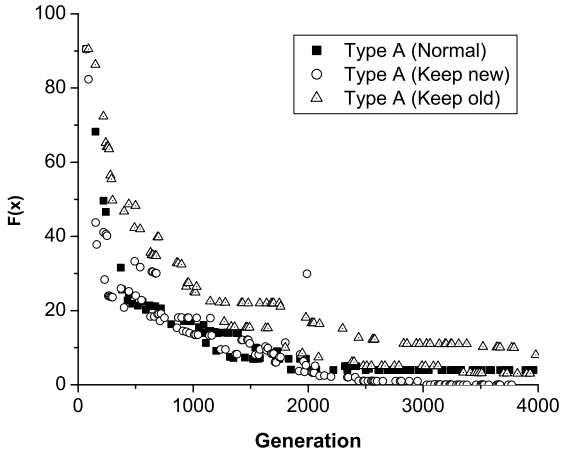

Figure 10: Effect of Asynchronous Migration

judge they can continue or restart their search from the view point of the reduction timing when it happens. There is higher risk in the initial search process and lower risk in the final process. The small amounts of reduction do not affect the results severely.

In Asynchronous Migration, it is necessary to care for the migration between the islands, when the evolution speed is different in each island. That influence was small in our experiments when the individuals are assigned to each computational resource equally. The arrangement that obtains the same evolution speed should be operated as possible. The immigrants from the island where the evolution speed is slow do not affect to the total search process. On the other hand, the immigrants from the island where the evolution speed is fast contribute a lot for the search process.

As these results, the following items are the guidelines for executing the DGA on the Grid.

1. Use resources as possible as users can. The higher accuracy of the results is expected when the more resources are used.

2. Users fix the population size or number of island for the certain CPU power. When users find out that they can use new resources, they add the population. In that case, the population size should have the same ratio as the CPU power of new resource has.

3. Allow the dynamic reduction when other users call for some resources if the initial search process has finished. The amount should be the least.

4. Asynchronous migration has the lower overhead for the network communication. The evolution speed in each island should be arranged the same. Controlling the population size of each island is one of these arrangements. The tiny difference does not bring serious poor performance. The migration from the island where the evolution speed is fast to the island where the evolution speed is slow is important.

7 Conclusions

In this paper, the basic discussions of DGA model that is performed on the Grid have been performed. When the DGA is performed on the Grid, it is supposed that users can use huge computational resources and some of the resources are disappeared during the search. These discussions are performed only on three types of the test functions.

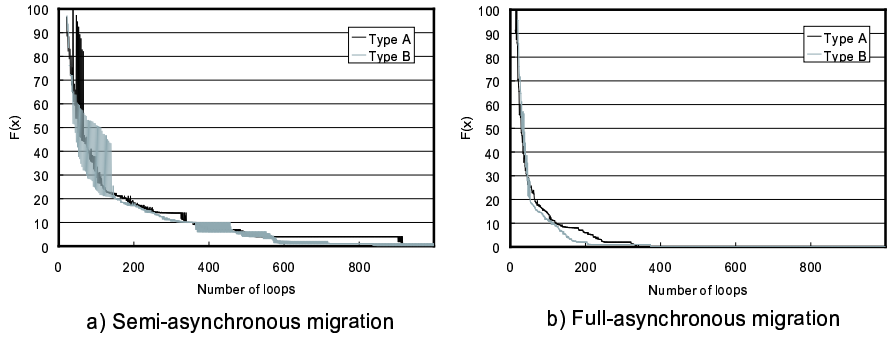

Figure 8: Comparison with Type A and Type B (Time step)

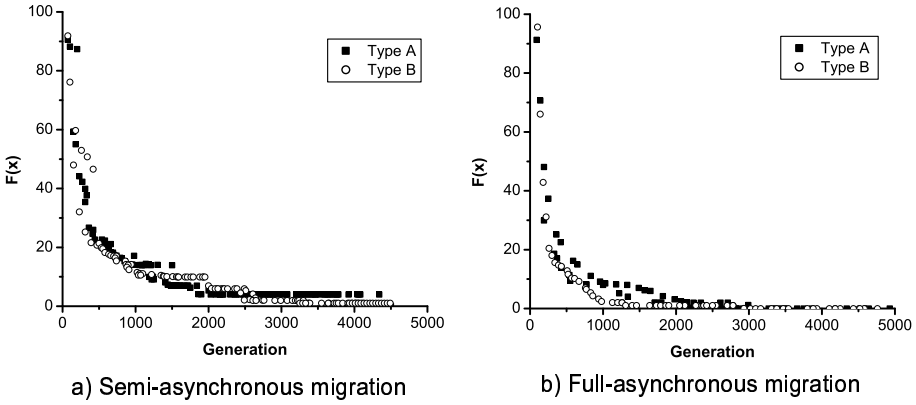

Figure 9: Comparison with Type A and Type B (Search step)

However, some general tendencies of the search behavior of GA on the Grid can be obtained. From the experiments of this paper, it is found that the DGA can be extended to the model where the computational resources are changing dynamically during the search. At the same time, the asynchronous migration for the Grid is also discussed. The DGA model has almost the same performance even if the evolution speed is different. In this case, the migration from the island where the evolution speed is fast contributes more to the search process. Through the experiments and discussions, the guideline of performing the DGA on the Grid is described. Some points can be applied to another extensions of the DGA model.

Acknowledgments

This work was supported by a grant to RCAST at Doshisha University from the Ministry of Education, Science.

Bibliography

[foster98] Foster, I. and Kesselman, J.M. (1998) “The Grid: Blueprint for a New Computing Infrastructure,” Morgan Kaufmann.

[sc00casanova] Casanova, H., Obertelli, G., Berman, F. and Wolski, R. (2000) “The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid,” Proc. of the Supercomputing(SC).

[hcw00shao] Shao, G., Berman, F. and Wolski, R. (2000) “Master/Slave Computing on the Grid,” Proc. of the Heterogeneous Computing Workshop.

[sahpc97casanova] Casanova, H. and Dongarra, J. (1997) “NetSolve: A Network Server for Solve Computational Science Problems,” *International Journal of Supercomputer Application and High Performance Computing*, Vol.11, No.3.

[grid02nakada] Nakada, H., Matsuoka, S., Seymour, K., Dongarra, J., Lee, C. and Casanova, H. (2002) “GridRPC: A Remote Procedure Call API for Grid Computing,” Proc. of Grid Workshop.

[anl00goux] Goux, J., Linderot, J. and Yoder, M. (2000) “Metacomputing and the Master-Worker Paradigm,” ANL/MCS-P792-0200.

[goldberg89] Goldberg, D.E. (1989) “Genetic Algorithms in Search, Optimization, and Machine Learning,” Addison-Wesley.

[comp99alba] Alba, E. and Troya J. M. (1999) “A survey of Parallel Distributed Genetic Algorithms,” *Complexity*, Vol.4, No.4.

[cantu-paz00] Cantú-Paz, E. (2000) “Efficient and Accurate Paralell Genetic Algorithms,” *Genetic Algorithms and Evolutionary Computation*, Vol.1, Kluwer Academic Pub.

[eaecs99tomassini] Tomassini, M. (1999) “Parallel and Distributed Evolutionary Algorithms: A Review,” *In Evolutionary Algorithms in Engineering and Computer Science*, J. Wiley and Sons.

[pc91muhlenbein] Mühlenbein, H., Schomisch, D. and Born, J. (1991) “The Parallel Genetic Algorithms as Function Optimize,” *Parallel Computing*, Vol.17, No.6–7.

[um75dejong] De Jong, K.A. (1975) “Analysis of the Behaviour of a Class of Genetic Adaptive Systems,” PhD thesis, University of Michigan.

[rechenberg94] Rechenberg, I. (1994) “Evolutionstrategie ’94,” Frommann-Holzboog Verlag.

[seal02sano] Sano, M., Hiroyasu, T. and Miki, M. (2002) “Discussion of Search Phases of Probabilistic Model-building Genetic Algorithms,” Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning(SEAL), pp.472–476.

[ga2kspec3] Hiroyasu, T., Yoshida, J., Sano, M., Fukunaga, T. and Kataura, T., (2002) “Distributed Genetic Algorithms ga2k (Ver.1.3) Specification,” ISDL, Doshisha University, http://mikilab.doshisha.ac.jp/dia/research/pdga/archive/20020925_ver1.3e/ga2k_doc_E2.pdf.

[ipsj02hiroyasu] Sano, M., Hiroyasu, T. , Miki, M., and Kamiura, J. (2002) “ A Presumption of Parameter Settings for Distributed Genetic Algorithms by Using Design of Experiments,” IPSJ Journal, Vol. 43, No. SIG 10(TOM 7),

[pdcs02tanimura] Tanimura, Y., Hiroyasu, T. and Miki, M. (2002) “The System for Evolutionary Computing on The Grid,” Proc. of IASTED Parallel and Distributed Computing and Systems.