World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:2, No:6, 2008

Asynchronous Parallel Distributed Genetic Algorithm with Elite Migration

Kazunori Kojima, Masaaki Ishigame, Goutam Chakraborty, Hiroshi Hatsuo, and Shozo Makino

Abstract**—** In most of the popular implementation of Parallel GAs the whole population is divided into a set of subpopulations, each subpopulation executes GA independently and some individuals are migrated at fixed intervals on a ring topology. In these studies, the migrations usually occur 'synchronously' among subpopulations. Therefore, CPUs are not used efficiently and the communication do not occur efficiently either. A few studies tried asynchronous migration but it is hard to implement and setting proper parameter values is difficult.

The aim of our research is to develop a migration method which is easy to implement, which is easy to set parameter values, and which reduces communication traffic. In this paper, we propose a traffic reduction method for the Asynchronous Parallel Distributed GA by migration of elites only. This is a Server-Client model. Every client executes GA on a subpopulation and sends an elite information to the server. The server manages the elite information of each client and the migrations occur according to the evolution of sub-population in a client. This facilitates the reduction in communication traffic.

To evaluate our proposed model, we apply it to many function optimization problems. We confirm that our proposed method performs as well as current methods, the communication traffic is less, and setting of the parameters are much easier.

Keywords**—** Parallel Distributed Genetic Algorithm (PDGA), asynchronous PDGA, Server-Client configuration, Elite Migration

I. Introduction

GENETIC Algorithm (GA) [1] is a multi-point search technique imitating the survival of the fittest rule of nature. The approximate solutions are obtained by repeating the selection, the crossover and the mutation operations called genetic operations, on a set of approximate solutions. An individual in the population has the chromosome information which is an encoded solution of the problem.

Since J. H. Holland et al. proposed GA in the early 1970s, GA had been successfully applied to various function optimization problems, combinational optimization problems, control problems, machine learning and so on [2]. This is because GA is easy to implement, it is robust, and it could find global optimum solution.

In recent years, it has been required to get the better approximate solutions faster and more efficiently for larger and more complicated problems. However, it is hard to obtain the approximate solution for these problems by using GA running on a single machine, because it is computationally heavy.

There are two basic approaches to make GA faster. In approach one, the algorithm itself is modified from its generic model to a more specific algorithm to solve the problem at hand, or hybridize GA with artificial neural network, simulated annealing or some other heuristics.

The second approach is to speed up the computation by parallel distributed implementation (PDGA). Many researches in the PDGA are coarse-grained parallel GA which is implemented as follows. The population is divided into some subpopulations and each subpopulation is assigned to a different processor. GA is executed in parallel on each processor. To prevent premature convergence of the subpopulation, some individuals from each processor are exchanged (migrated) among different subpopulations.

There are two main methods to migrate individuals. One is the synchronous method where migration occurs at fixed generation intervals. In this method, a fast CPU needs to wait for the response from a slow CPU for synchronization of migration. Therefore, it is not efficient for CPU utilization especially in a heterogeneous environment. The other is the asynchronous method where individuals are migrated with fixed probability. But it is hard to implement the asynchronous event in general. Furthermore, the migrations are not driven by the searching need in both the methods.

To solve these problems, Munetomo et al. [10] proposed a migration method that controls the migration timings by the difference of the standard deviation in a particular subpopulation. However, it is very hard to set the parameter to control the migration timing.

In this paper, we propose a migration method, we named Elite Migration. This is an Asynchronous Parallel Distributed Genetic Algorithm implementation on a Server-Client topology. In this method, a client station executes GA independently on a subpopulation and send an elite chromosome information to the server when an elite is updated. The server manages elite information received from each subpopulation. When an elite of a client is not updated for several generations, the server sends some manipulated chromosome information to the client and the client receives the chromosome information as migrants. Therefore, migration is less and controlled by the server.

To evaluate the proposed method, we applied our method to find the global optimum for some uni-modal and multi-modal functions, and confirmed that the results are as good as obtained by other methods with much less message passing.

II. Parallel Distributed GA

Researches on PDGA can be classified roughly into 4 approaches [3].

Kazunori K., Masaaki I. and Goutam C. are with the Iwate Prefectural University, Iwate, Japan, 020-0193. E-mail: kojima@iwate-pu.ac.jp. Hiroshi M. is with the Tohoku Seikatsu Bunka College. Shozo M. is with the Tohoku University.

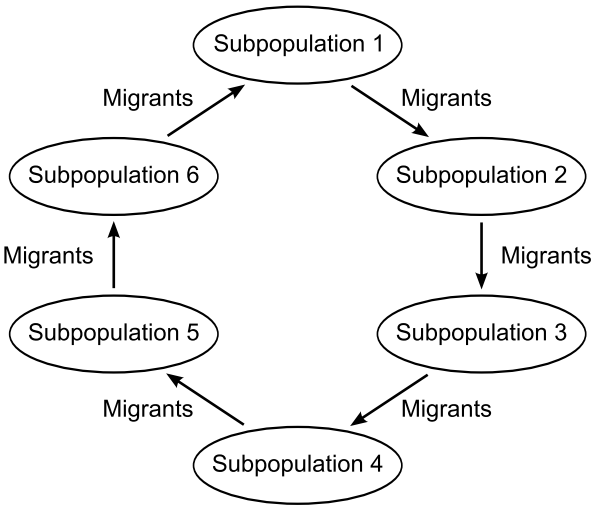

Fig. 1. Island model: migrants are sent synchronously in fixed intervals on a ring topology

1) Global Parallelization [4], [5], where the whole population is assigned to a single processor and the calculations of fitness or the genetic operations are assigned to different processors.
2) Coarse-grained Parallelization [6], [7], [8], [9], [10], where the population is divided into some subpopulations and the individuals are exchanged between these subpopulations.
3) Fine-grained Parallelization [11], [12], [13], where individuals are assigned on a grid structure and an individual can crossover with its neighbouring individuals.
4) Works that combine the above [14], [15].

Among these researches, the subpopulation based methods [6], [7], [8], [9], [10] are the most natural parallelization method and can be implemented easily.

In PDGA using subpopulation based methods, the island model is usually used. In the island model, the population is divided into some subpopulations and each subpopulation executes GA independently. To prevent convergence of each subpopulation, individuals need to be exchanged between subpopulations at fixed intervals or with fixed probability.

In most of the island models, ring topology is used as shown in Figure 1. Individuals are sent to next island and are received from the previous island synchronously at fixed intervals. Hereafter, we call this model Synchronous model. In the Synchronous model, a fast processor has to wait for a slow processor for the migration to be synchronized. Therefore, in an heterogeneous environment, where there are some slow processors, the available computation power can not be used efficiently.

On the other hand, there is a model where individuals are exchanged with fixed probability. In this model, each subpopulation requests migration asynchronously. Hereafter, we call this model Random-Exchange model. However, in general, it is hard to process such asynchronous events. Furthermore, from the viewpoint of genetic search, it is not efficient if migration occurs either randomly or at fixed intervals, because migrations are not motivated by necessity.

To solve the above problems, Munetomo et al. proposed Sigma-Exchange model [10]. In this model, the standard deviation of fitness in each subpopulation is observed. When fitness of members in a subpopulation converges, migration occurs. However, it is hard to set the parameter that controls migration because we need to set the parameter according to the characteristics of the application and/or the population size.

Recently Erick et al. [3] reported various researches and experiments with PDGA. However, they didn't discuss about the communication traffic.

Adachi et al. proposed Parameter-free GA [16], [17] and Hiroyasu et al. proposed Dual Individual Distributed Genetic Algorithm [18]. They implemented their algorithm with Master-Slave configuration. However, the migrants pass between the master and the slave synchronously, creating heavy communication load. They did not discuss about the communication traffic either.

Thus two main problems of PDGA are reducing unnecessary migration which involves communication cost, and setting different parameters for maximum efficiency. To overcome these problems, we propose Asynchronous elite migration PDGA as explained in section III.

III. Asynchronous Elite Migration PDGA

The model we used in this work forms a Server-Client configuration as shown in Figure 2, composed of a server program and some client programs. Hereafter, we call our proposed model as APDGA-EM (Asynchronous Parallel Distributed Genetic Algorithm with Elite Migration).

Each client program executes GA independently with a subpopulation. The server program being executed as Elite Server manages the elite information of each subpopulation. Hereafter, we call the server as Elite Server, and client as Subpopulation Client.

This Elite Server-Subpopulation Client model has two merits. It is easy to manage asynchronous event, and it does not affect the results even if some Subpopulation Client break down. Moreover, it is possible to add new Subpopulation Client to the system or delete useless Subpopulation Client from the system as the genetic search progresses.

Elite Server communicates to each Subpopulation Client according to the following rules.

- When an elite is updated in a Subpopulation Client, the Subpopulation Client sends the elite chromosome information to Elite Server.
- Each Subpopulation Client has a Longevity parameter. When the genetic search does not go well in a Subpopulation Client, in other words an elite is not updated, Longevity parameter (λ) is decremented.
- When Longevity parameter (λ) is 0 with a Subpopulation Client, it requests migration to Elite Server as shown in Figure 3. Then, that Subpopulation Client receives new chromosome information from Elite Server.
- Elite Server receives elite information from each Subpopulation Client and sort them by fitness order. When a Subpopulation Client requests migration, Elite Server

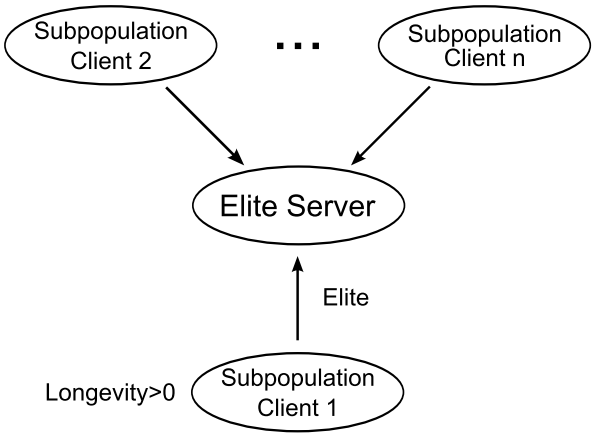

Fig. 2. Asynchronous PDGA with Elite Migration: an elite chromosome information is sent to Elite Server when an elite in a Subpopulation Client is updated

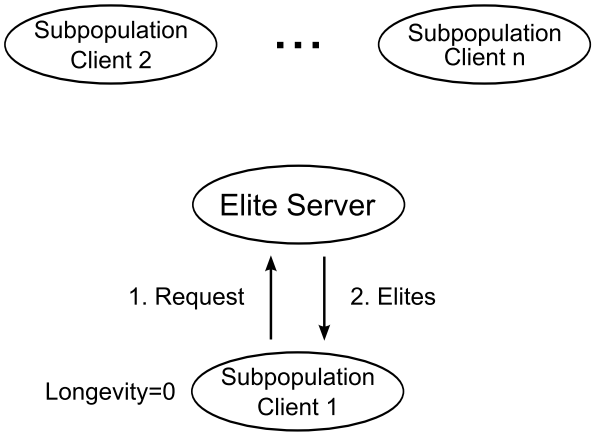

Fig. 3. Asynchronous Migration between Elite Server and Subpopulation 1: A Subpopulation Client requests migration to Elite Server and receives μ of elite chromosome information when Longevity parameter is 0

sends μ number of chromosomes from the top of the list, i.e., the μ best chromosomes Elite Server knows.

The communication in Elite Migration consists of sending an elite chromosome information from a Subpopulation Client to Elite Server, sending a migration request from a Subpopulation Client to Elite Server and as response, a Subpopulation Client receives the chromosome information from Elite Server. They are asynchronous events because they occur at various times. Communication occurs when an elite in a Subpopulation Client is updated. So unnecessary communication is reduced and the communication traffic is less.

A. Parameters

There are three parameters in Elite Migration.

1) Interval to Send an Elite: A Subpopulation Client does not send an elite chromosome information at every elite updating, but once in every ν times of updates. By increasing ν, our system is able to reduce the communication frequency from a Subpopulation Client to Elite Server. We need to optimize this interval ν.

2) Longevity Parameter: The Longevity parameter (λ) is to control the timing of the request for chromosome import from Elite Server. To simplify the implementation, λ is decremented by 1 when, in a generation, an elite is not updated. And the migration occurs when λ is decremented to 0. When an elite is updated, λ is reset to the initial value. Thus, when GA searches well within a Subpopulation Client, it does not need to send migration request to Elite Server and the communication frequency is low.

3) The Number of Migrants: As described before, the migration occurs when Longevity Parameter in a Subpopulation Client is decremented to 0. The Subpopulation Client requests to Elite Server for migrants and Elite Server sends μ chromosome information to the Subpopulation Client. Individuals selected randomly in the Subpopulation Client are replaced by newly received individuals. The ensuing traffic could be kept low when the number of migrants (μ) is small.

B. Implementation

We implemented the simulation using C and MPI, and executed on SGI Origin2000. Origin2000 is a parallel processing machine with 32 CPUs. MPI (Message Passing Interface) [19] is the standard library for message communication. MPI is used on parallel computers but there are similar packages for UNIX or WindowsNT, so that one can use MPI on WSs or PCs on a TCP/IP network.

When we implement Parallel GA on WSs or PCs on a TCP/IP network, we can also use PVM (Parallel Virtual Machine), Socket library, Java language and so on. It is hard to implement Random-Exchange and Sigma-Exchange using the above techniques. However, it is easy to implement our algorithm using them.

Figure 4 shows Problem Analysis Diagram (PAD) of Elite Server program that we used, and Figure 5 shows the PAD of the Subpopulation Client program.

Elite Server waits for messages from Subpopulation Clients until they exist. When a message arrives, Elite Server processes it according to its content and then waits for further messages. In our programs, messages are identified using TAG of MPI. Messages can be identified by including TAG in the message on Socket programming or Java language.

The Subpopulation Client executes GA and sends an elite information once every ν times the elite in that island is updated. When an elite is not updated, Longevity parameter (λ) is decremented. If Longevity parameter is 0, the Subpopulation Client sends a request to Elite Server for migration and receives migrants from Elite Server.

A message is composed of a TAG and the message itself. The TAG is used to identify the message. The main body of the message mainly is the chromosome itself. Thus, a message that a Subpopulation Client sends to the Elite Server is the chromosome information of the elite in that Subpopulation Client. A message that a Subpopulation Client receives from the Elite Server is some chromosome information that the Elite Server manages.

Figure 4 and Figure 5 omit the details. But it is evident from these figures that the model is easy to implement and that the communication cost is low.

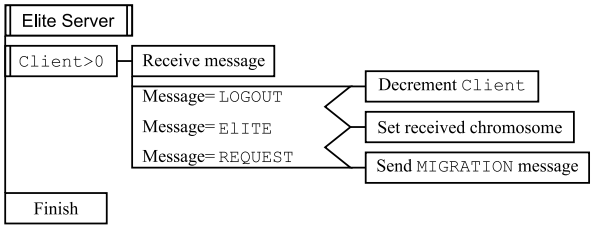

Fig. 4. PAD of Elite Server: Elite Server processes only messages from Subpopulation Clients

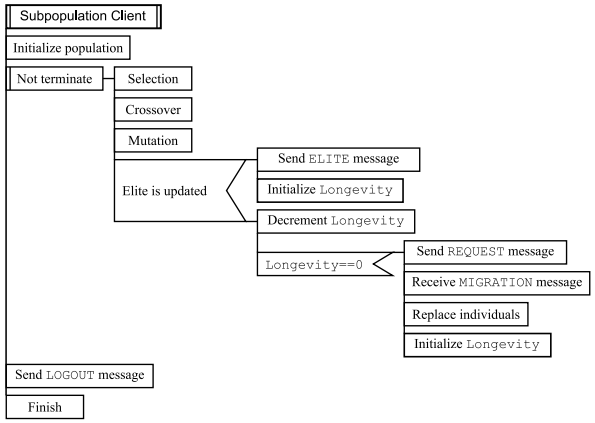

Fig. 5. PAD of Subpopulation Client: Subpopulation Client executes GA and processes messages according to an elite status

IV. Experiments and Results

In this section, we describe the experiments and the results. For comparison, we implemented Synchronous model, Random-Exchange, Sigma-Exchange and APDGA-EM on Origin2000.

A. Experimental conditions

We experimented the above mentioned algorithms on 8 function optimization problems. Table I shows the list of functions *Func1* to *Func8*, we experimented with.

Func1 and *Func2* are a n-dimensional uni-modal function and a n-dimensional multi-modal function respectively. We created these two functions. Each variable is encoded to 10-bit gray code in the chromosome. *Func3* and *Func4* are De Jong's Function F1 and F5. *Func5*, *Func6*, *Func7* and *Func8* are the functions that were used in the first International Contest on Evolutionary Optimization (ICEO) in 1996 and the second ICEO in 1997. From *Func3* to *Func8*, the precision of a solution is set to 6 decimal places and each variable is encoded to 22–31 bits gray code. The optimization criterion is to get maximum value for all the functions.

The parameters for GA are as follows. Total population size is 1024. Selection is roulette selection. Crossover is single point crossover for each variable. Crossover rate is 0.6. Mutation is applied at bit level with 0.03 probability. An elite is preserved at every generation and it is terminated at the 500th generation.

The parameters for PDGA are follows. Migration rate is 0.2. Migration interval is 100 generations in Synchronous model. Migration probability is 0.01 in Random-Exchange. From preliminary experiments of Sigma-Exchange a suitable value for the parameter to control migration timing is selected. And ring topology is used.

In the proposed APDGA-EM, interval ν to send an elite is set at 10, initial Longevity λ is 100 and the number of migrants is 2.

TABLE I
List of application problems

Func1: n-Half-Sine
$f(x_i) = \frac{1}{n}\sum_{i=1}^{n} \sin\left(\frac{\pi x_i}{1024}\right)$
$n = 5, 0 \le x_i \le 1023$, chromlen=$10 \times n$
Func2: n-Sine-Cosine
$f(x_i) = \frac{1}{n}\sum_{i=1}^{n} \left(0.5 \sin\left(\frac{\pi x_i}{1024}\right) \cos\left(\frac{20\pi x_i}{1024}\right) + 0.5\right)$
$n = 5, 0 \le x_i \le 1023$, chromlen=$10 \times n$
Func3: De Jong's Function F1 (Parabola)
$f(x_i) = -\sum_{i=1}^{3} x_i^2$
$-5.12 \le x_i \le 5.12$, chromlen=24×3
Func4: De Jong's Function F5 (Shekel's foxholes)
$f(x_i) = -\left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$
$-65.536 \le x_i \le 65.536$, chromlen=$27 \times 2$
Func5: Generalized Langerman's function
$f(x_i) = \sum_{i=1}^{m} c_i e^{-\frac{1}{\pi} g(x_i)} \cos(\pi g(x_i))$
$g(x_i) = \sum_{j=1}^{n} (x_j - a_{ij})^2$
$m = 30, n = 5, 0 \le x_j \le 10$, chromlen=$24 \times n$
Func6: Sphere model
$f(x_i) = -\sum_{i=1}^{n} (x_i - 1)^2$
$n = 5, -5 \le x_i \le 5$, chromlen=$24 \times n$
Func7: Griewank's function
$f(x_i) = -\left(\frac{1}{4000}\sum_{i-1}^{n} g(x_i)^2 - \prod_{i=1}^{n} \cos\left(\frac{g(x_i)}{\sqrt{i}}\right) + 1\right)$
$g(x_i) = x_i - 100$
$n = 5, -600 \le x_i \le 600$, chromlen=$31 \times n$
Func8: Michalewicz' function
$f(x_i) = \sum_{i=1}^{n} \sin(x_i) \sin^{2m}\left(\frac{i x_i^2}{\pi}\right)$
$m = 10, n = 5, 0 \le x_i \le \pi$, chromlen=$22 \times n$

B. Comparison of fitness and traffic

In this set of experiments, the number of islands is 16 and each algorithm was executed for 25 times.

Table II shows the mean of best fitness till termination. In this table, Sync, Rand-ex and Sig-ex indicate Synchronous model, Random-Exchange model and Sigma-Exchange model respectively.

From this table, we can confirm that the fitnesses obtained by APDGA-EM are as good as obtained by other methods.

TABLE II
COMPARISON OF MEAN OF THE BEST FITNESS

	Sync	Rand-ex	Sig-ex	APDGA-EOM
Uni-modal	9.9999E-01	9.9999E-01	9.9999E-01	9.9999E-01
Multi-modal	9.9418E-01	9.9468E-01	9.9444E-01	9.9546E-01
De Jong F1	-1.9696E-08	-2.2352E-08	-2.5914E-08	-3.2331E-08
De Jong F5	-9.9800E-01	-9.9800E-01	-9.9800E-01	-9.9800E-01
Langerman	9.8969E-01	9.8654E-01	9.8728E-01	9.9337E-01
Sphere	-1.1312E-05	-9.8701E-06	-1.6630E-05	-1.4850E-05
Griewank	-3.1375E+02	-3.0876E+02	-3.1379E+02	-3.0879E+02
Michalewicz	4.6875E+00	4.6876E+00	4.6875E+00	4.6875E+00

Therefore, in spite of lower communication cost, which we will show later, APDGA-EM could deliver at least equally good results.

Table III shows the mean of the traffic that one Subpopulation Client has received in the simulations. Table IV shows the mean of the traffic that one Subpopulation Client has sent to Elite Server in the simulations

TABLE III
COMPARISON OF RECEIVED DATA AT SUBPOPULATION CLIENT (BYTES)

	Sync	Rand-ex	Sig-ex	APDGA-EOM
Uni-modal	1200.00	2317.60	410.24	41.60
Multi-modal	1200.00	2279.04	324.16	40.00
De Jong F1	720.00	1357.60	208.80	6.72
De Jong F5	480.00	837.12	0.00	3.84
Langerman	1200.00	2269.92	139.52	0.00
Sphere	1200.00	2441.76	362.40	3.20
Griewank	1200.00	2201.76	348.96	1.60
Michalewicz	1200.00	2480.96	54.24	4.80

TABLE IV
COMPARISON OF SENT DATA BY SUBPOPULATION CLIENT (BYTES)

	Sync	Rand-ex	Sig-ex	APDGA-EOM
Uni-modal	1200.00	2313.60	408.32	76.96
Multi-modal	1200.00	2275.20	322.56	66.40
De Jong F1	720.00	1359.36	204.48	69.76
De Jong F5	480.00	837.12	0.00	46.08
Langerman	1200.00	2284.80	132.16	142.40
Sphere	1200.00	2448.00	359.04	126.72
Griewank	1200.00	2208.00	343.68	140.96
Michalewicz	1200.00	2486.40	50.56	116.48

The traffic of received data is the same as the one of sent data in Synchronous model because the number of message passing is same. The traffic in Random-Exchange is almost twice as that of Synchronous model, and here too the traffic of received data is almost same as the sent data. In Sigma-Exchange, both the received data traffic and sent data traffic is less than the ones in Synchronous model and Random-Exchange. Convergence of population in De John's Function F5 (*Func4*) is difficult, because the difference between higher fitness value and lower fitness value is large. Therefore, migration takes place only occasionally and the traffic is less. Also in Michalewicz Function (*Func8*), migration happens seldom and the traffic is less. In these two functions, the traffic of the received data is almost same as the sent data.

In APDGA-EM, the traffic is more than that of Sigma-Exchange for some functions. But the traffic is less than other methods in general. The traffic of the received data are the migrants from Elite Server and the traffic of the sent data are the elites sent to Elite Server. Therefore, these two traffics are different. When an elite in a Subpopulation Client is updated frequently, the traffic of sent data increases. On the other hand, the decrementing of Longevity Parameter (λ) rarely happens. Therefore, the migration does not occur frequently and the traffic of received data is less.

From the results of our experiments, we can confirm that the traffic in APDGA-EM is less compared to other methods. And still it could achieve comparable chromosome fitnesses. Therefore, APDGA-EM is a more efficient parallel algorithm.

C. Parameters of APDGA-EM

In APDGA-EM, there are the following three parameters.

- Initial value of Longevity Parameter (λ)
- The number of elite for migrants (μ)
- Interval to send an elite information to Elite Server (ν)

Among these parameters, initial value of Longevity Parameter (λ) and the number of elite for migrants (μ) affect the communication cost for received data, and interval (ν) to send an elite information affects the communication cost for sent data.

Figure 6 shows the dependence of final fitness on initial Longevity Parameter. Figure 7 shows the dependence of final fitness on the number of migrants. Figure 8 shows the variation of final fitness on the interval to send an elite information.

From Figure 6 and Figure 7, we can confirm a tendency of negative and positive gradient respectively. For example, when the initial value of Longevity Parameter is small or when the number of migrants is large, the fitness is better. In summary, this means that the fitness is better when more messages are passed between Elite Server and Subpopulation Client, and it is worse when the traffic is less. However, the fitness fluctuates within a narrow range. Therefore, we can also confirm that these parameters do not affect the fitness strongly.

From Figure 8, it is seen that the resulting fitness does not really depend on the interval at which elite is passed from Subpopulation Client to Elite Server, and the fitness varies within a very narrow range. Similar behaviour is confirmed for all other functions too. This is an important difference from the parameter of Sigma-Exchange that has to be changed according to population size, GA operators and the application

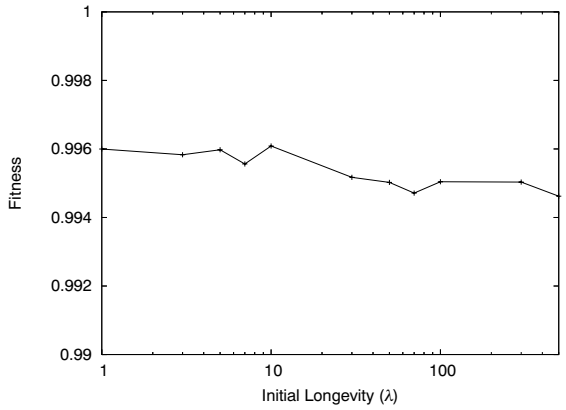

Fig. 6. Dependence on initial Longevity parameter: Fitness is better when Initial Longevity (λ) is small value

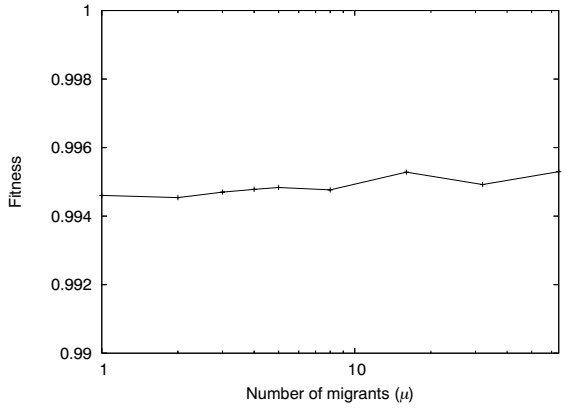

Fig. 7. Dependence on the number of migrants: Fitness is better when the number of migrants (μ) is large value

problem. We thus conclude that the proposed algorithm is more robust to the selection of the parameters for various applications.

From all the experiments, we recommend the general optimum values for the parameters as:

- Initial Longevity parameter (λ) to be 100
- The number of elite information to send for migrants (μ) to be 2
- Interval to send an elite information to Elite Server (ν) to be 10

The above setting could deliver near optimum values for all the experiments we performed.

V. Conclusion

In this paper, we proposed Elite Migration for Asynchronous Parallel Distributed Genetic Algorithm which uses Server-Client topology. Our proposed method mainly sets some rules to communicate between Elite Server and Subpopulation Client which run genetic operations on a small population of chromosomes. A Subpopulation Client sends an elite chromosome information to Elite Server when an elite is updated. To control migration timing, Longevity Parameter

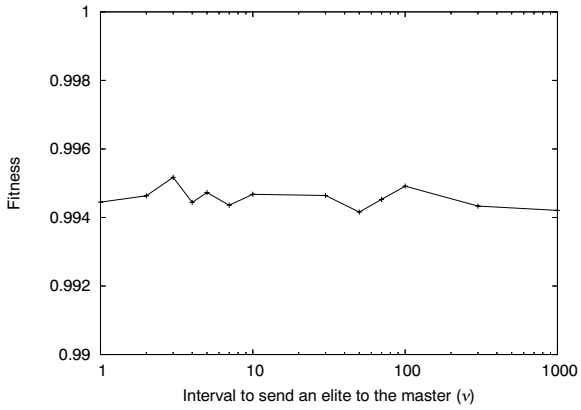

Fig. 8. Dependence on interval to send an elite: Fitness does not really depend on this value

(λ) is introduced on each Subpopulation Client. And some chromosome information that the Elite Server manages are used for migrants. Thus the proposed algorithm reduces the message passing traffic in PDGA.

We applied our proposed method and other PDGA methods to various function optimization problems and confirmed the effectiveness of our proposal. The traffic in proposed method is less than other current PDGA methods, and the results by our proposed method is better or at least equally good as those obtained by other methods.

Three parameters of the proposed method could affect the fitness values and the generated traffic. However, we have shown that the effect of the parameter values are only marginal and do not need to be tuned for different problems. Thus the proposed algorithm is robust to variations in parameter values, and the same set of values work well for every problem.

Therefore, the proposed algorithm is robust as well as its communication cost is lower compared to its competitive algorithms

Acknowledgments

The authors would like to thank Iwate Prefectural University Media Center Administrative staffs, who made possible the access to the computation environment needed to pursue this work.

References

[1] Holland, J.H.: "Adaption in Natural and Artificial Systems", University of Michigan Press (1975).
[2] Goldberg, D.E.: "Genetic Algorithm in Search Optimization and Machine Learning", Addison Wesley (1989).
[3] Erick Cantú-Paz: "Efficient and Accurate Parallel Genetic Algorithms", Kluwer Academic publishers (2000)
[4] Fogarty, T.C., and Huang, R.: "Implementing the genetic algorithm on transputer based parallel processing systems", Parallel Problem Solving from Nature, pp.145-149 (1991).
[5] Hauser, R., and Männer, R.: "Implementation of standard genetic algorithm on MIMD machines" Parallel Problem Solving from Nature, PPSN III, pp.504-513 (1994)
[6] J.P. Cohoon, W.N. Martin, and D.S. Richards: "A Multi-population Genetic Algorithm for Solving the k-Partition Problem on Hypercubes", Proc. of ICGA-91, pp.244-248 (1991).

[7] C.C. Petty, and M.R. Leuze: "Theoretical Investigation of a Parallel Genetic Algorithm", Proc. of ICGA-89, pp.398-405 (1989).
[8] R. Tanese: "Parallel Genetic Algorithm for a Hypercube", Proc. of ICGA-87, pp.177-183 (1987).
[9] R. Tanese: "Distributed Genetic Algorithms", Proc. of ICGA-89, pp.434-439 (1989).
[10] Munetomo M., Yoshiaki T., and Yoshiharu S., "An Efficient Sigma Exchange Algorithm for a Subpopulation-Based Asynchronously Parallel Genetic Algorithm and Its Evaluation", IPSJ, vol.35, no.9, pp. 1815-1827, 1994.
[11] R.J. Collins, and D.R. Jefferson: "Selection in Massively Parallel Genetic Algorithms", Proc. of ICGA-91, pp.249-256 (1991).
[12] B. Manderick, and P. Spiessens: "Fine-grained Parallel Genetic Algorithms", Proc. of ICGA-89, pp.428-433 (1989).
[13] P. Spiessens, and B. Manderick: "A Massively Parallel Genetic Algorithm, Implementation and First Analysis", Proc. of ICGA-91, pp.279-285 (1991).
[14] D.E. Brown, C.L. Huntley, and A.R. Spillane: "A Parallel Genetic Heuristics for the Quadratic Assignment Problem", Proc. of ICGA-89, pp.406-415 (1989).
[15] M. Gorges-Schleuter: "ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy", Proc. of ICGA-89, pp.422-427 (1989).
[16] Sachio K., Hidefumi S., and Susumu A.: "Parameter-free Genetic Algorithm (PfGA) Using Adaptive Search with Variable-Size Local Population and Its Extension to Parallel Distributed Processing", IEICE Transactions(D-II), Vol.J82-D-II, No.3, pp.512-521 (1999).
[17] Susumu A., and Hidefumi S.: "Effects of Migration Methods in Parallel Distributed Parameter-Free Genetic Algorithm", IEICE Transactions(D-I), Vol.J83-D-I, No.8, pp.834-843 (2000).
[18] Tomoyuki H., Mitsunori M., Masahiro H., and Yusuke T.: "A New Model of Distributed Genetic Algorithm for Cluster Systems: Dual Individual DGA", Proc. of PDPTA, Vol.1, pp.477-483 (2000).
[19] P. Pacheco: "Parallel Programming with MPI", Baifu-kan (2001)

Kazunori Kojima received the B.Eng. and M.Eng. degree in mining college from Akita University in 1993 and 1995, respectively. He is currently a research associate of Faculty of Software & Information Science of Iwate Prefectural University. He has been engaged in research on genetic algorithms.

Masaaki Ishigame received the Ph.D. degree in Tohoku University in 1974. He was a research associate of Tohoku University, an employee of Matsushita Denso, and an associate professor of Akita University. He is currently Dean of the graduate school of Iwate Prefectural University. He has been engaged in research on signal processing, image processing, and knowledge engineering.

Goutam Chakraborty received his Ph.D. in 1993 from Tohoku University, Japan. Presently he is Professor and head of the Intelligent Informatics lab., Department of the Software and Information Science, Iwate Prefectural University, Japan. His main research interests are Soft Computing algorithms and their applications to solve pattern recognition, prediction, scheduling and optimization problems including applications in wired and wireless Networking problems.

Hiroshi Matsuo received the Ph.D. degree in Tohoku University in 1990. He was an employee of Canon, and a research associate and an assistant professor of Akita University. He is currently an associate professor of Tohoku Seikatsu Bunka College.

Shozo Makino received the Ph.D. degree in Tohoku University in 1974. He was a research associate, and an associate professor of Tohoku University. He is currently a professor of Tohoku University. He has been engaged in research on voice recognition and understanding, voice database, voice signal processing, voice CALL system, image processing, character recognition.