Parallel Distributed Processing of a Parameter-free GA by Using Hierarchical Migration Methods

Hidefumi Sawai and Susumu Adachi
Kansai Advanced Research Center,
Communications Research Laboratory, MPT,
588-2, Iwaoka, Iwaoka-cho, Nishi-ku, Kobe, 651-2401 Japan,
e-mail: {sawai, sadachi}@crl.go.jp, Phone number: +81 78 969 2160

Abstract

Parallel distributed processing of parameter-free genetic algorithm (PfGA), which is free from the need to set genetic parameters such as initial populations size, crossover, and mutation rates in advance, has been developed. The PfGA is a compact and robust algorithm that extracts a minimum local population by focusing on a promising part of a population and evolves it by adaptively changing the size of the local population. Parallel distributed architectures of a PfGA with hierarchical immigration methods are realized in a parallel machine. To evaluate the performance of the parallel distributed architectures, we used function optimization problems as an international benchmark test. We demonstrate that the average number of evaluations needed to successfully reach the global optima decreases gradually and the success rate becomes 100% by increasing the number of local populations.

1 INTRODUCTION

The genetic algorithm (GA) (Holland 1975) is an evolutionary computation paradigm inspired by biological evolution. GAs have been successfully applied to many practical applications such as functional optimization problems, combinatorial optimization problems, and optimal design of parameters in machines (Goldberg 1989). However, the design of genetic parameters used in a GA has to be determined by trial and error, making optimization by GA *ad hoc*. One of the most important research areas in evolutionary computation is to adapt genetic parameters and operators in a *self-adaptive* manner because such adaptation can tune an algorithm while solving a given problem. In the literature (Hinterding 1997), a classification of adaptation is developed, which covers different levels (such as environment, population, individual, and component) and types (such as static and dynamic ones).

However, it is a very time-consuming task to design an optimal evolutionary strategy in an adaptive way because we have to perform the evolutionary algorithm many times by trial and error. To relieve the user of this kind of adaptive parameter-setting burden, we developed a parameter-free genetic algorithm (PfGA) (Kizu 1997, 1999; Sawai 1998), in which no control parameters for genetic operations need to be set as constants in advance. It merely uses arbitrarily random values or probabilities for setting almost all genetic parameters. The PfGA is inspired by the *disparity theory of evolution* proposed by Furusawa et al. (Furusawa 1992; Wada 1993). Their idea is based on the disparity of copy error rates in the *leading* and *lagging* strands of DNA when each strand makes its copy. This leads to *diversity* in a biological ecosystem. The search strategy in the PfGA is based on a dynamic change of subpopulation size extracted from the population. This strategy enables an adaptive search to take a delicate balance between global and local search methods. These two kinds of search methods correspond to exploration and exploitation, respectively, and maintain diversity in GAs.

We have performed parallel distributed processing of the PfGA in a synchronous mode. Two kinds of architectures with three different hierarchical migration methods are realized in a parallel machine. The performances achieved with these different parallel processing methods are compared by using the five benchmark functions from an international contest on evolutionary optimization, ICEO'96 (Bersini 1996).

2 PARAMETER-FREE GA INSPIRED BY THE DISPARITY THEORY OF EVOLUTION

2.1 DISPARITY THEORY OF EVOLUTION

As Charles Darwin claimed in the "Origin of Species"

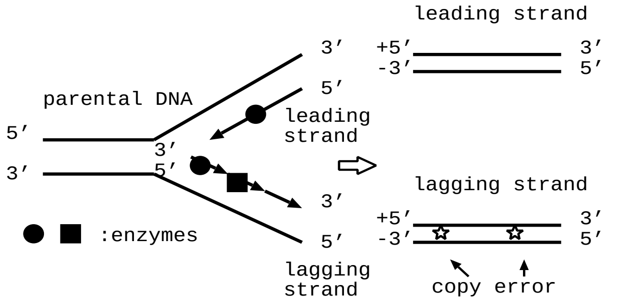

Figure 1: A hypothesis in the *disparity theory of evolution*

(Darwin 1859), a major factor contributing to evolution is mutation, which can be caused by spontaneous misreading of bases during DNA synthesis. Semiconservative replication of double-stranded DNA is an asymmetric process in which there is a *leading* and a *lagging strand*. Furusawa et al. proposed a *disparity theory of evolution* (Furusawa 1992) based on a difference between the frequencies of strand-specific base misreading of leading and lagging DNA strands (i.e., disparity model). Figure 1 shows a hypothesis in the disparity theory of evolution. In the figure, the leading strand is copied smoothly; whereas in the lagging strand a copy error can occur because plural enzymes are necessary to produce its copy. This disparity or asymmetry in producing each strand occurs because of the different mutation rates in the leading and lagging strands. Thus "diversity" of DNAs is maintained in a population as generations proceed. The disparity model guarantees that the mutation rate of some leading strands is zero or very small. When circumstances change for example, when the original wild type can not survive, selected mutants might adapt under the new circumstances as a new wild type. In their study (Wada 1993), the disparity model was compared with a parity model in which there was no statistical difference between the frequencies of base misreading of strands as in the generally accepted model. The disparity model outperformed the parity model in solving a knapsack optimization problem (Wada 1993).

2.2 PARAMETER-FREE GENETIC ALGORITHM

The population of the PfGA is considered as a whole set S of individuals which corresponds to all possible solutions. From this whole set S, a subset S' is introduced. All genetic operations such as selection, crossover, and mutation are conducted for S', thus evolving the subpopulation S'. From the subpopulation S', we introduce a *family* which contains two parents and two children generated from the two parents (see Figure 2 (left)).

The PfGA procedure is as follows:

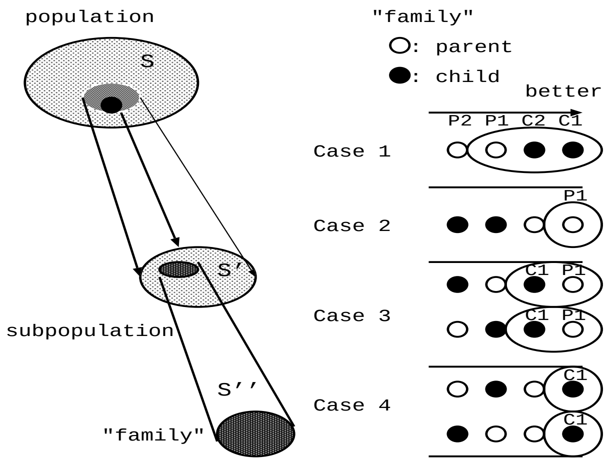

Figure 2: Population S, subpopulation S', family S'' (left), and selection rules (right) in a parameter-free GA

Step 1. Select one individual randomly from the whole population S and add this individual to the subpopulation S'.

Step 2. Select one individual randomly from the whole population S and again add this individual to the subpopulation S'.

Step 3. Select two individuals randomly from the subpopulation S' and perform crossover between these individuals as "parent 1 (P_1)" and "parent 2 (P_2)".

Step 4. For one randomly chosen child of the two children generated from the crossover, perform mutation at random.

Step 5. Among the parents (P_1 and P_2) and the two generated children (C_1 and C_2) select one to three individuals depending on the following cases (i.e., cases 1 to 4 to follow) and feed them back to the subpopulation S'.

Step 6. If the number of individuals in subpopulation S' is greater than one, go to Step 3; otherwise, go to Step 2.

For the crossover operation of PfGA, we use multiple-point crossover in which n crossover points (n is a random number, $0 < n <$ gene length, which changes every time the crossover takes place) are randomly selected and genes between two parents' chromosomes are replaced. For the mutation operation, one child is randomly chosen from the two offsprings. Then a randomly chosen portion of the child's chromosome is inverted (i.e., bit-flipped). For the selection operation, we compare the fitness values of all individuals (C_1, C_2, P_1, and P_2) in the family. The selection rules shown in Fig. 2 (right) are used for four different cases depending on the fitness values f of the parents and children.

Case 1: If the fitness values of the two children are better than those of the parents, then C_1, C_2 and $arg\ max_{P_i}(f(P_1), f(P_2))$ are left in S', thus increasing the size of S' by one.

Case 2: If the fitness values of C_1 and C_2 are worse than those of P_1 and P_2, then only $arg\ max_{P_i}(f(P_1), f(P_2))$ is left in S', thus decreasing the size of S' by one.

Case 3: If the fitness value of either P_1 or P_2 is better than that of the children, $arg\ max_{C_i}(f(C_1), f(C_2))$ and $arg\ max_{P_i}(f(P_1), f(P_2))$ are left in S', thus maintaining the size of S'.

Case 4: In all other situations, $arg\ max_{C_i}$ $(f(C_1), f(C_2))$ is preserved and then one individual randomly chosen from S is added to S', thus maintaining the size of S'. A more detailed description is provided elsewhere (Sawai 1998; Kizu 1999).

3 PARALLEL DISTRIBUTED PROCESSING OF A PFGA WITH HIERARCHICAL MIGRATION METHODS

Generally speaking, parallel processing aims at accelerating the speed of processing. In the case of a GA, it aims at reaching better solutions faster than sequential processing by extending the search space. Parallel and distributed processing of a GA has been extensively studied (Belding 1995; Evans 1998; Maruyama 1993). The *granularity* ranges from fine- to coarse-grained and the mode of processing covers both synchronous and asynchronous processing, where the *granularity* concerns the size of a process assigned to a processor. In the case of a fine-grained parallel GA model, the (overlapping) neighborhoods of the individuals constitute the units of processing. On the other hand, a coarse-grained parallel GA assigns a subpopulation as a unit of processing, and a few individuals are migrated among subpopulations at an appropriate rate. The latter model is called an "island model", and one island (subpopulation) consists of one "deme" which is the minimum recombinational unit of a biological species. In this paper, we use the latter model.

Figure 3 shows an architecture of a parallel PfGA. This architecture is based on that shown in Figure 2. The whole population S is located at the center, from which N subpopulations $\{S'_i\}$ $(i = 1, \ldots, N)$ are extracted ($N = 8$, in this case). Of course, if the number of processors increases, the number of subpopulations could also increase. Each *family* shown as S''_i is extracted from each subpopulation S'_i. If better individuals are produced in a subpopulation S'_i, the individuals are copied and migrated among the subpopulations (the bilateral arrows indicate this situation). One possible migration method is as follows: If case 1 or 4 occurs in

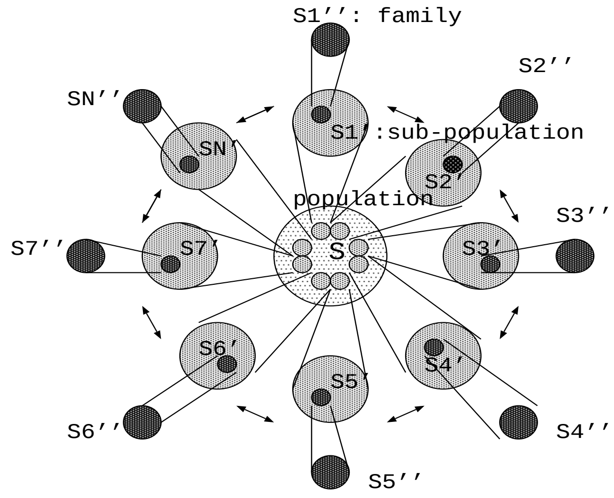

Figure 3: A parallel distributed architecture of the PfGA

some family, the individual C_1 that is better than its two parents can be copied to other subpopulations as an emigrant. In case 1, since the two children are better than their two parents, it might be possible for the two children to emigrate to other subpopulations, but this might cause a premature convergence decreasing the diversity of the population (in fact, case 1 rarely occurs). When other subpopulations receive the immigrant, they have to decide to accept it or not because the number of individuals will increase and lead to an explosion of diversity if any immigrants are unconditionally accepted. To avoid such a situation, one possible method is to eliminate the worst individual among all individuals including the "candidate" immigrant in the subpopulation. This maintains the sizes of subpopulations. If the immigrant is eliminated, the immigration cannot be substantially realized.

Figure 4 shows another type of architecture of a parallel PfGA. This is called a "master-slave" type of paralled distributed PfGA. Let S'_0 be the master subpopulation and $\{S'_i\}$, $(i = 1, \ldots, N)$ be the slave subpopulations. The master S'_0 always or at some interval supervises the subpopulations $\{S'_i\}$ and identifies the best individual among them. One possible migration method is as follows; When case 1 or 4 occurs in a subpopulation $\{S'_i\}$, compare C_1 with the best-so-far individual identified by the master subpopulation. If the former (C_1) is better than the latter (actually, this situation always occurs), copy and distribute it to all other subpopulations. In each subpopulation that receives the "candidate" immigrant, the worst individual among all individuals including the candidate immigrant is eliminated from the subpopulation. This elimination maintains the number of indiduals in the subpopulation as in the uniformly distributed architecture.

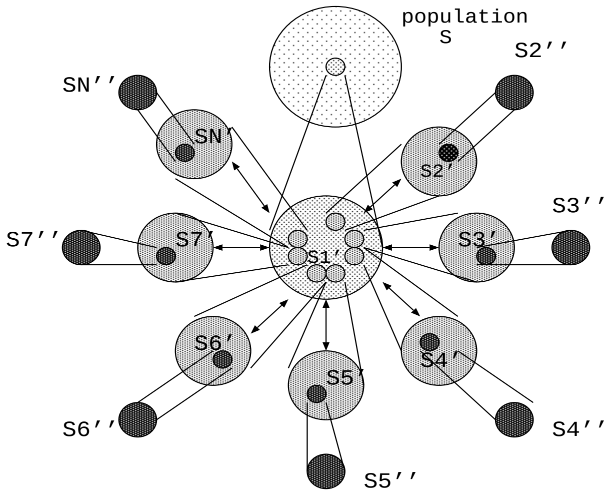

Figure 4: A master-slave type architecture of the PfGA

Table 1: Test functions

$F1 : Sphere Model :$
$f(x_i) = \sum_{i=1}^{n}(x_i - 1)^2, \quad -5 \leq x_i \leq 5$
$F2 : Griewank's Funtion :$
$f(x_i) = \frac{1}{d}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n} cos(\frac{x_i - 100}{\sqrt{i}}) + 1,$
$d = 4000, -600 \leq x_i \leq 600$
$F3 : Shekel's\ foxholes :$
$f(x_i) = -\sum_{i=1}^{m}\frac{1}{\sum_{j=1}^{n}(x_j - a_{ij})^2 + c_i},$
$m = 30, 0 \leq x_i \leq 10$
$F4 : Michalewicz' Function :$
$f(x_i) = -\sum_{i=1}^{n} sin(x_i) sin^{2m}(\frac{i x_i^2}{\pi}),$
$m = 10, 0 \leq x_i \leq \pi$
$F5 : Generalized\ Langerman's\ Function :$
$f(x_i) = -\sum_{i=1}^{m} c_i[exp(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j - a_{ij})^2)$
$cos(\pi \sum_{j=1}^{n}(x_j - a_{ij})^2)],$
$m = 5, 0 \leq x_i \leq 10$

We also propose the following three kinds of migration methods embedded in the two kinds of architectures:

- Uniformly distributed architecture with all migration (**UD0**):
  If case 1 or 4 occurs in S'_i, $(i = 1, \ldots, N)$, copy C_1 and distribute it to all other subpopulations $\{S'_j\}$ $(j \neq i, j = 1, \ldots, N)$.
- Uniformly distributed type with unilateral migration (**UD1**):
  In the above UD0, copy C_1 and distribute it to a subpopulation S'_j, $(j \neq i)$. All other operations are the same as in the UD0.
- Uniformly distributed architecture with hierarchical migration (**UD2**):
  In the above UD1, two parents, P_1 and P_2, in S'_j are operated by crossover and mutation, thus producing two children, C_1 and C_2. Four individuals are selected according to the selection rule of a PfGA, and one, two, or three individuals are distributed to other subpopulations.
- Master-slave architecture with all migration (**MS1**):
  S'_0 denotes the master subpopulation and S'_i, $(i = 1, \ldots, N)$ denotes slave subpopulation. If case 1 or 4 occurs in S'_i, C_1 is distributed to the master S'_0. The master selects the best individual among the distributed ones from the slaves, then it copies and distributes it to all other slave subpopulations.
- Master-slave architecture with hierarchical migration (**MS2**):
  In the above MS1, the master selects two parents, P_1 and P_2, that are distributed from a slave subpopulation. The two parents are operated by crossover and mutation, thus producing two children, C_1 and C_2. Four individuals are selected according to the selection rule of a PfGA, and one, two, or three individuals are distributed to other slave subpopulations.

These architectures were implemented by using several processors connected with local-area networks (LAN). PVM (parallel virtual machine) (Geist 1993) software is used to evolve the subpopulations in parallel and synchronously. PVM is network software that uses dynamic load balancing between assigned processors.

4 EXPERIMENT

Experiments on solving the function optimization (minimization) problems listed in Table 1 were performed. The dimension n in all the problems was set to five. These functions have their own upper and lower bounds defined by their range of variables x_i (Baeck 1997) and are explained as follows: $F1$, the Sphere model, which takes its minimum 0 at $x_i = 1$, is separable and is a shifted function of De Jong's Sphere model ($f(x) = \sum_{i=1}^{n} x_i^2$, (De Jong 1975)); $F2$ is Griewank's function which takes its minimum 0 at $x_i = 100$; $F3$ represents Shekel's foxholes (De Jong 1975) with m local minima. Its global minimum is approximately -10.40; $F4$ is Michalewicz' function which takes its global minimum, -4.688; and the last function, $F5$, the Generalized Langerman's function, has a global minimum of -1.4 or below, which is very difficult to find by conventional optimization methods such as neural networks or simulated annealing. Depending on each function, the defined variables were encoded using 22 to 24 bits in Gray coding.

To evaluate the performance of the PfGA with that of other evolutionary algorithms, we compared it with the results of the first ICEO (International Contest on Evolutionary Optimization) (Bersini 1996). Bersini et al. held the first ICEO at the 1996 ICEC (Int. Conf. on Evolutionary Computation), at which eight different algorithms took part. In the functional optimization problems, three performance indexes were introduced; expected number of evaluations per success (ENES), best value reached (BV), and relative time (RT). A 5-dimensional version of the test bed was used to measure each of these indexes. The ENES index represents the mean number of function evaluations needed to reach a certain fitness value, value to reach (VTR), given with each problem. ENES is computed by running 20 independent runs of the algorithm until the VTR is reached. And RT is defined by (CT-ET)/ET, where CT is the total CPU time to perform the algorithm with 10,000 iterations, and ET is the CPU time to perform 10,000 evaluations of the fitness function. Consequently, the lower RT, the more compact the algorithm becomes.

The PfGA took second place in tests of the 5-dimensional version among nine algorithms including itself. Since its RT is nearly one, the PfGA is more compact than the others. We also measured success rate and ENES when using the PfGA for parallel distributed processing. For $N = 2, 4, 8, 16, 32, 64$, and 128, three hundred trials from different random seeds were run. For $N = 1$ (single processing) only, ten thousand trials were run in order to get enough precision. For UD2 only, N gradually increased to reach a success rate of 100%.

5 EXPERIMENTAL RESULTS

The five cases — two different architectures and three kinds of migration methods — are compared. The results are shown in Figs. 5 to 9, where the horizontal axis represents the number of subpopulations N and the vertical axis represents the success rates R or ENES.

In the Sphere model, the success rates reached 100% and BV reached zero in all cases except UD0. And ENES decreased as $1/N$ when N increased (Fig. 5). Though it is difficult to determine which architecture is the best according to the results only, MS1 is probably the best when N is more than 64, followed by MS2, UD1, UD2, and UD0. When N is more than 16, ENES in UD1, UD2 and MS2 saturated. In the case of UD0, the calculation was terminated when N reached 16 because of increasing computational load of migration of $O(N^2)$.

In the Griewank function, although the success rate of 0.23% at $N = 1$ was extremely low, it gradually increased as N increased (Fig. 6). In this case it is also difficult to order the results, but we can roughly

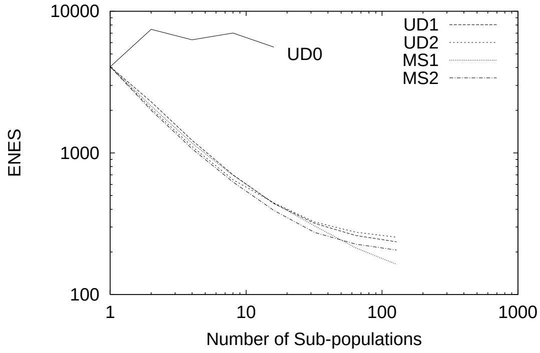

Figure 5: ENES vs N in terms of Sphere Model

say that the most increased case is UD2, followed by MS2, MS1 and UD1. The success rate was zero in the case of UD0. But in the case of UD2, it was confirmed that the success rate reached 100% when $N = 2048$. In regards to ENES, because the success times are very short, the average of ENES statistically fluctuates. If the upper limit of evaluations increases beyond 10,000 times, it is easy to imagine that the success rate will increase and ENES will also increase.

Regarding Shekel's foxholes, although the success rate was only 7.06% at $N = 1$, it increased as N increased (Fig. 7), and for UD2 at $N = 512$, it was confirmed that it reached 100%. The success rates of UD0 and MS1 became lower than others. In terms of ENES, when N is more than 16, UD2 is the most efficient, followed by UD1, MS2, MS1, and UD0.

In terms of Michalewicz' function, the success rate was 44.34% at $N = 1$. It reached 100% at an N of more than 32 in all cases except UD0, and ENES followed a $1/N$ curve at N more than 16 (Fig. 8). The success rates similarly increased except in the case of UD0. In terms of ENES, UD0 is the least efficient, MS1 tended to converge slowly when N is more than 16, and the other architectures showed similar performance.

Regarding Langerman's function, the success rate was 32.32% at $N = 1$, and it increased except in the case of UD0 as N increased (Fig. 9). In terms of success rates at $N = 128$, UD1 is the best, followed by UD2, MS2, MS1, and UD0. It was confirmed that the success rate achieved by UD2 reached 100% at $N = 256$. In terms of ENES, UD0 is the most difficult to converge, followed by MS1. And UD1, UD2, and MS2 are almost the same in terms of convergence.

According to the above results as a whole, the success rates and ENES achieved by UD0 and MS1 are relatively inferior to those of the other cases. This is because both UD0 and MS1 distribute the best individual to all subpopulations, and this loses diversity in the population. This situation makes all subpopulations search for the same points with the same "best"

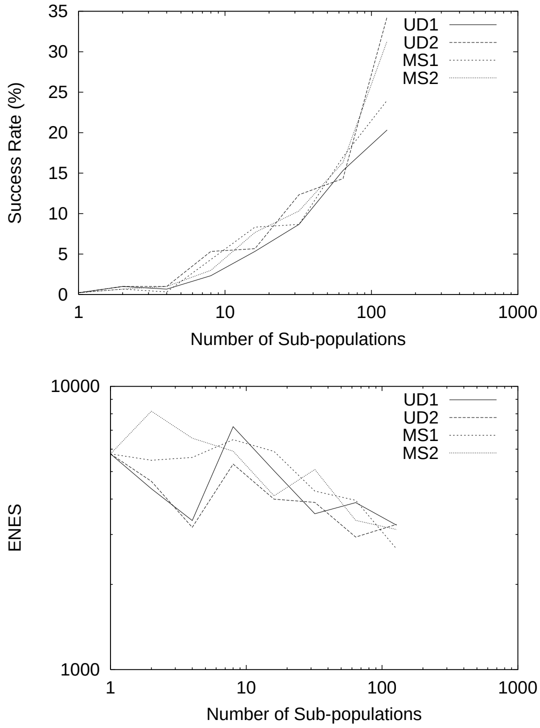

Figure 6: Success rates and ENES v.s. N in terms of Griewank's Function

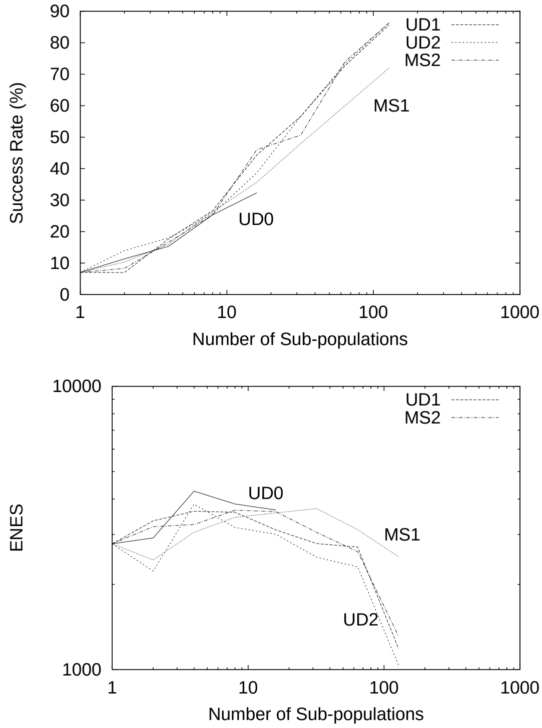

Figure 7: Success rates and ENES v.s. N in terms of Shekel's foxholes

individual. However, in terms of Sphere model, ENES showed the best performance in the case of MS1. The reason is that as the number of local minima is one and the global optimum at the same time, the best individual identified by the master is always the best-so-far individual that is closest to the global optimum.

6 DISCUSSION

In the Sphere model, for example, the probability occurring the case 1 or 4 in one trial, i.e., one occurring the case that produces the better individual C_1, is 15% at $N = 1$. This probability varies as a function of N as shown in Fig. 10. It decreases after a peak at $N = 2$ in the two types of master-slave architectures, MS1 and MS2, and conversely it increases in the two uniformly distributed architectures, UD1 and UD2. It converges to 18.8% in the cases of the two hierarchical migration methods, UD2 and MS2. This kind of behavior for the other functions is qualitatively the same when the probability converges to a constant value. As a result, the number of migrated individuals is proportional to N and especially it becomes the same in the case of two hierarchical migration methods, UD2 and MS2, when N increases.

If no migration occurs among subpopulations, ENES is constant (ENES at $N = 1$) and independent of the number of subpopulations because each subpopulation evolves separately. R_N denotes the success rate in the case of no migration and R_1 denotes the success rate at $N = 1$. If at least one subpopulation in N subpopulations reaches VTR, it can be recognized as a successful trial. Therefore, R_N is represented using R_1 as

$$R_N = 1 - (1 - R_1)^N.$$

This curve for five functions as well as the success rates when using UD2 is shown in Fig. 11. Note that the R_N curve is not an upper limit of the success rate when migration occurs, because the success rate achieved by UD2 applied to Griewank's function exceeds the R_N curve.

We can consider ENES from the following viewpoints. Generally speaking, when we apply parallel processing with N processors to an optimization problem, it might be an easy method that if the exploration space could be equally divided into N subspace, each processor can explore each subspace independently. This method makes it possible to reduce the iterations by $1/N$ because the number of points to explore decreases to $1/N$. Nevertheless, the parallel PfGA does not divide the exploration space into N subspaces in an artificial way, ENES decrease as $1/N$ when using UD1, UD2, and MS2 in the cases of the Sphere model at N less than 16, Michalewicz' function at N more than 16, and Shekel's function at N more than 64 (the case of

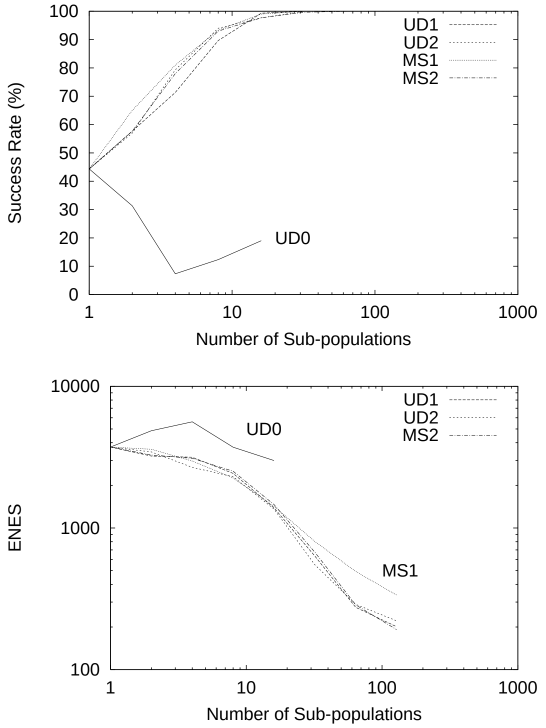

Figure 8: Success rates and ENES v.s. N in terms of Michalewicz' Function

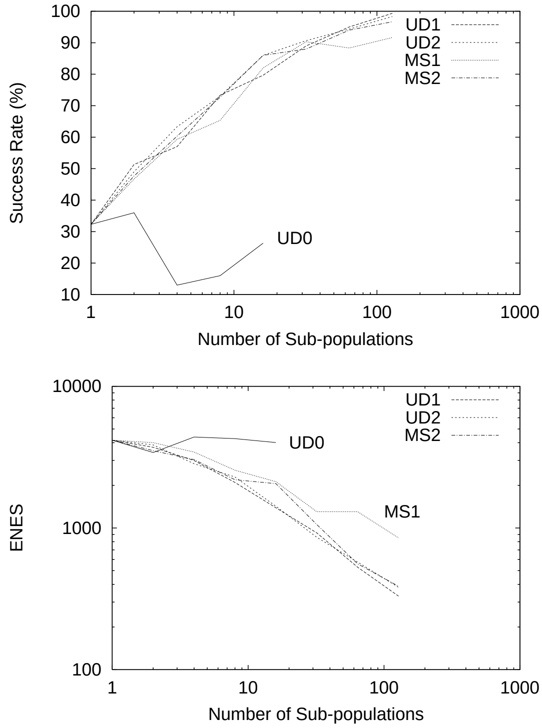

Figure 9: Success rate and ENES v.s. N in terms of Langerman's Function

UD2 is shown in Figure 12). It can be interpreted that each subpopulation of these three architectures ideally and dynamically explores the search space, maintaining the diversity by taking a balance not to overlap each other.

7 Summary

Parallel distributed processing of a parameter-free genetic algorithm (PfGA), which is free from the need to set genetic parameters such as initial populations size, crossover and mutation rates in advance, is described. The PfGA is a compact and robust algorithm that extracts a minimum local population by focusing on a promising part of a population and evolves it by adaptively changing the size of the local population. Parallel distributed architectures of PfGA with hierarchical immigration methods are realized in a parallel machine. For evaluating the performance of parallel distributed architectures, we used function optimization problems as an international benchmark test. We demonstrated that the average number of evaluations needed to successfully reach the global optima decreases gradually and the success rate becomes 100% by increasing the number of local populations.

Furthermore, this synchronously parallel processing can be sequentially implemented using a single processor even though the processing speed decreases as $1/M$ (M is the number of parallel processors). Future study will focus on its applications to real-world problems and on an artificial ecosystem in a changeable environment using parallel processing of a PfGA.

Acknowledgements

The authors would like to express their gratitude to Mr.S. Kizu for providing the PfGA program with a subpopulation and to Dr.S. Maekawa for his useful discussions.

References

Charles Darwin (1859), "On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life," London, John Murray.

J. H. Holland (1975), "Adaptation in Natural and Artificial System," The University of Michigan Press.

D. E. Goldberg (1989), "Genetic Algorithm in Search, Optimization, and Machine Learning," Addison Wesley.

De Jong (1975), "An Analysis of the Behavior: a Class of Genetic Adaptive Systems," Doctoral dissertation, University of Michigan.

M. Furusawa and H. Doi (1992), "Promotion of Evolution: Disparity in the Frequency of Strand-specific Misreading Between the Lagging and Leading DNA Strands Enhances

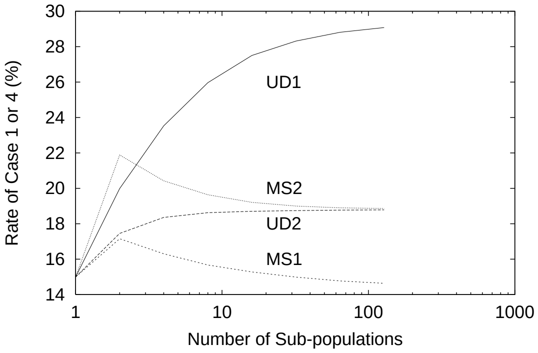

Figure 10: Rate that Case 1 or 4 occurs in terms of Sphere model

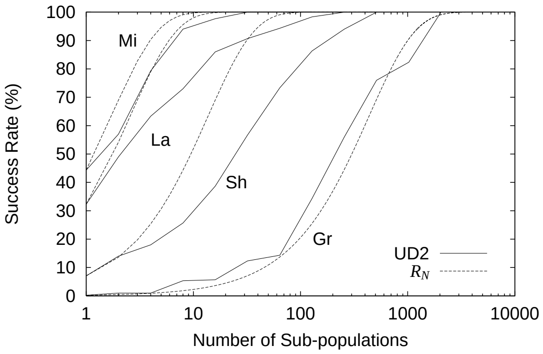

Figure 11: Success rates in the case of UD2 and R_N v.s. number of sub-populations

Disproportionate Accumulation of Mutations," J. Theor. Biol., vol. 157, pp 127-133.

K. Wada, H. Doi, S. Tanaka, Y. Wada, and M. Furusawa (1993), "A Neo-Darwinian Algorithm: Asymmetrical Mutations due to Semiconservative DNA-type replication Promote Evolution," Proc. Natl. Acad. Sci., USA, vol. 90, pp 11,934-11,938.

T. Baeck, D. Fogel and Z. Michalewicz (1997) (Eds.), "Handbook of evolutionary computation," New York: Oxford University Press.

The Organising Committee: H. Bersini, M. Doringo, S. Langerman, G. Seront, L. Gambardella (1996), "Results of the First International Contest on Evolutionary Optimization (1st ICEO)," 1996 IEEE Int. Conf. on Evolutionary Computation (ICEC'96), pp 611-615.

Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam (1993), "PVM UER'S GUIDE AND REFERENCE MANUAL."

R. Hinterding, Z. Michalewicz, and A. E. Eiben (1997), "Adaptation in Evolutionary Computation: A Survey," Proc. of the 1997 IEEE Int. Conf. on Evolutionary Computation, pp 65-69.

T. C. Belding (1995), "The Distributed Genetic Algorithm

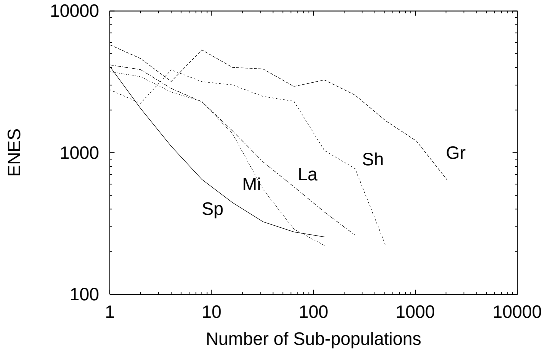

Figure 12: ENES v.s. N in the case of UD2

Revisited," Proc. of the Sixth Int. Conf. on Genetic Algorithms, pp 114-121, Morgan Kaufmann.

I. K. Evans (1998), "Embracing Premature Convergence: The Hypergamous Parallel Genetic Algorithm," Proc. of the Int. Conf. on Evolutionary Computation '98, pp 621-626.

T. Maruyama, T. Hirose and A. Konagaya (1993), "A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel System," Proc. of the Fifth Int. Conf. on Genetic Algorithms, pp 184-190, Morgan Kaufmann.

S. Kizu, H. Sawai, and T. Endo (1997), "Parameter-free Genetic Algorithm: GA without Setting Genetic Parameters," Proc. of the 1997 Int. Symp. on Nonlinear Theory and its Applications, vol.2 of 2, pp 1273-1276.

H. Sawai and S. Kizu (1998), "Parameter-free Genetic Algorithm Inspired by "Disparity Theory of Evolution"", Proc. of the Int. Conf. on Parallel Problem Solving from Nature, pp 702-711.

S. Kizu, H. Sawai and S. Adachi (1999), "Parameter-free Genetic Algorithm (PfGA) Using Adaptive Search with Variable-size Local Population and its Extension to Parallel Distributed Processing," Trans. IEICE, Japan, to be pubulished.

Bibliography

Hidefumi Sawai finished his doctor course in Keio Univ. in 1982. He was a senior researcher at ATR-ITL Lab. in Kyoto from 1988 to 1991, and an invited research scientist at CMU, PA, USA from 1989 to 1990. Currently, he is the Head of AVIS at KARC in CRL, MPT, engaging in modeling of evolutionary mechanisms. He was a member of the Editorial Committee in IPSJ, the Steering/Program Committee of JTTAS, WCCI'94, ICSLP'94, ICGA'95, NNSP'96, ALIFE-V'96 etc. and is a member of the IEEE, IEICE, IPSJ, AI Society of Japan, JNNS, and ALife Networks.

Sususmu Adachi was graduated from his master-course in Hiroshima University in 1995. His major was to study on quantum devises. During his doctor course he joined AVIS section in KARC, MPT in 1998, and is doing research on parallel processing of GAs.