An Improvement of Database with Local Search Mechanisms for Genetic Algorithms in Large-Scale Computing Environments

Yoshiko Hanada
Graduate School of Engineering,
Doshisha University
610-0321 Kyoto, Japan
hanada@mikilab.doshisha.ac.jp

Tomoyuki Hiroyasu
Department of Engineering,
Doshisha University
610-0321 Kyoto, Japan
tomo@is.doshisha.ac.jp

Mitsunori Miki
Department of Engineering,
Doshisha University
610-0321 Kyoto, Japan
mmiki@mail.doshisha.ac.jp

Abstract- Recently, GA that uses large-scale computer systems comprised of massive processors has become feasible because of the emergence of super PC clusters and Grid computation environments. Mechanisms to use massive computation resources laconically and to search effectively are necessary if large-scale computer systems are available. In this study, a new GA-specific database with the local search mechanism to assure the scalability of search performances against the number of computing resources is proposed. Our database possesses information of searched space, and in addition, the local search for non-searched spaces is applied using individuals stored in the database. To embed our database in a GA enables us to comprehend the quantitative rate of a searched region during searches. Applying this local search, the searched space can be expanded linearly in accordance with the increase in computing resources and the exhaustive search is guaranteed under infinite computations. The features of the introduced GA are discussed with reference to several types of experiments. This method was applied to primitive functions and test functions of continuous optimization problems. Through such experiments, it was shown that our method ensures an effective exhaustive search and has almost the same performance as a conventional GA.

1 Introduction

Genetic Algorithms (GAs) are among the most effective approximation algorithms for optimization problems. Various mechanisms for improving GAs have been discussed. Minimal Generation Gap (MGG)[1] was proposed as a generation alternation model. Methods using Linkage Identification[2, 3], Real-coded GA[4], and Distributed GA[5] are other GAs with strong search capabilities. Restart mechanisms have also been applied to enhance the performance of GAs[6, 7, 8, 9]. However, application of a GA to solve optimization problems has the drawback that these algorithms incur large computing costs. One solution to this problem is to perform GAs in parallel. GAs are well suited to parallel processing environments due to their ability to search with multiple points and their tolerance for extinction of search points, and consequently GAs have found applications in large-scale computing[10, 11, 12, 13]. In recent decades, the remarkable improvements in computing capabilities have rendered some of the drawbacks regarding the computing costs of GAs unimportant. Furthermore, parallel processing is used to yield increases in performance of GAs. Due to the recent emergence of super PC clusters and Grid computation environments, such as PC Grid comprised of desktop machines for home use or offices, the number of computational calculation resources is increasing. Thus, large computing projects in fields related to evolutionary computing have become feasible. However, the application of GAs on large-scale computer systems composed of massive processors, i.e., Mega Processors, has the drawback that these algorithms lack scalability in their performance improvement in accordance with increases in available computing resources. Therefore, huge computing resources cannot be used effectively. This is caused by overlapping searches as described in the next section.

In this study, we introduce a GA-specific database that possesses information regarding the space that has been searched already to avoid overlapping searches and make effective use of computing resources in consideration of scalability of search performance when GAs use large computer resources. The targets of our research are: 1) to design a database to store searched individuals, 2) to introduce the Tabu · search mechanism including the searched region and restart strategies in the non-searched region using the database, and 3) to apply our method in large-scale computing systems. In this paper, we discuss 1) and 2). In our previous work, we proposed the expression of the searched region using schemata and a local search mechanism as a compression method to store large regions searched by several individuals. We then showed that the searched space could be expanded as the number of computing resources increased, enhancing accuracy and reliability by applying the database with the local search mechanism to a GA, and it was clear that the proposed method also showed superior performance with limited computing costs[14]. Nevertheless, in this work, there was the drawback that computing costs increased exponentially in accordance with generations. In this paper, we proposed a new database and local search based on our previous work.

2 Drawback of GAs in large-scale computing systems

Although they are well suited for parallel processing GAs lack scalability of improvement in their performance in accordance with increases in available computing resources when large amounts of computer resources are available.

The most common method used for conventional GAs with regard to increased resource use is to increase the population size. However, increasing the number of individuals cannot yield increases in performance of GAs because enlarging the population also increases the diversity of the solutions, i.e., the convergence speed becomes slow, and consequently the optimum solution is not derived rapidly when GAs use large amounts of computing resources. Another way to make use of huge computing resources is parallelization of multiple populations where GAs are run on each node independently. GAs are superior algorithms that can obtain satisfactory solutions in the early stages of the search with small computing costs. However, searches overlap within a population and among several populations due to convergence. In addition, multiple trials applying restart strategies may converge to the same solution.

Fig. 1 shows the increases in the searched region against increases in number of trials of the GA in the 2-dimensional Rastrigin function. In this figure, the abscissa indicates the number of trials or computation nodes, the left ordinate is the rate of searched regions, and right ordinate is the total number of evaluations. In this experiment, a single variable was expressed as strings of 10 bits, and thus the scale of the whole search space was 2^{20}(=1.05 x 10^6).

The ER model[15] was used as the alternation in each generation. We applied a GA with uniform crossover and each couple, or parents, generated 20 children by crossover. The mutation rate was 0.05 (=1/L, where L is the chromosome length) and the population size was 10. The termination of each generation was set to 20 and 2010 evaluations were achieved. Thus, approximately 1.0x10^6 evaluations could be performed using 500 nodes, which was worth the exhaustive search.

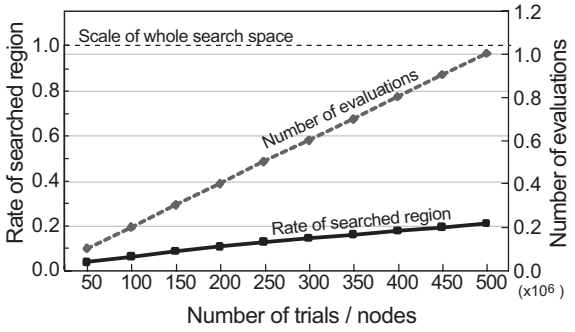

Figure 1: Increases in searched region against the number of evaluations

Fig. 1 illustrates that the rate of the region searched did not increase in comparison with increases in number of evaluations. In addition, only 20% had been searched although nearly exhaustive searches had been performed. This resulted in a waste of calculations.

Another way is to divide the whole search space into a number of segments, and the GA then runs on each segment independently. The segment number is increased in accordance with increases in available computing resources. Therefore, the possibility of overlapping increases due to segmentalization of the search space.

Due to these drawbacks, GAs cannot be utilized effectively on large-scale computer systems composed of massive processors because of the overlapping of searches.

3 Database Structure for Storing Individuals

In this study, we introduce a GA-specific database that carries information regarding the regions that have already been searched to avoid search overlap. When a database stores all the individual information, it takes a large time to check an individual that has already been searched due to the vast amounts of data. All search regions should be stored as highly compressed expressions and checking of individuals stored in the database should not be time-consuming. In this section, we describe the expression of the searched region.

3.1 Expression of Searched Regions

We used binary-coded individuals, and an individual or set of individuals is represented by 2-dimensional coordinates using the mapping method proposed by Collins, which converts the multidimensional search space to a 2-dimensional plane[16]. In this mapping method, an individual is treated as the coordinates (x, y), where the integer x is coded by Gray-coding from the string composed of genes of $2k$-th loci extracted from the chromosome, and integer y is coded from the string of extracted (2k+1)-th loci. Each individual has a one to one correspondence with a set of coordinates. Hence, the 2-dimensional plane can express the whole space. An example of representation of an individual by this mapping is shown in Fig. 2.

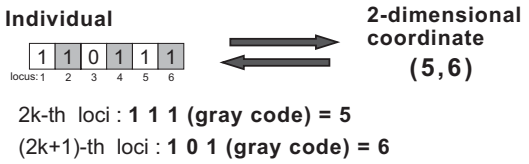

Figure 2: Example of coding an individual to 2-dimensional coordinates

This method is based on Gray-coding and the Hamming distances between neighbors are always 1 at each point. Thus, proximity on the 2-dimensional plane can locally correspond to the proximity between binary strings, and consequently structurally similar individuals can be treated as a contiguous region. Therefore, this method is appropriate to store all searched individuals in a small database.

In our proposed database, the set of individuals that have contiguous coordinates on the 2-dimensional plane are represented by a rectangle and are stored by two diagonal vertexes (x_{min}, y_{min}), (x_{max}, y_{max}) of the rectangle. In addition, the best individual in the searched region is stored.

Fig. 3 illustrates the whole search space of a problem consisting of 6 bits represented as a 2-dimensional plane and examples for a set of individuals represented as rectangles. Fig. 4 shows the database that possesses searched regions shown in Fig. 3. In this figure, a rectangle is repre-

sented with (x_{min}, y_{min})-(x_{max}, y_{max}), which are diagonal vertexes.

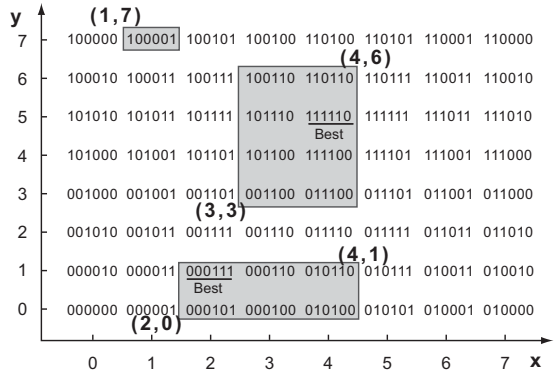

Figure 3: An Example of a 6-bit Problem Space created by the Mapping

	Searched region	Best individual	Size of searched region
data1	(3,3)-(4,6)	111110	8 (4x2)
data2	(2,0)-(4,1)	000111	6 (2x3)
data3	(1,7)-(1,7)	100001	1 (1x1)

Figure 4: Database representing searched regions illustrated in Fig. 3

This notation enables us to comprehend the quantitative rate of the searched region by calculating the square measure of the rectangle.

4 Operations in the Database

Our proposed database possesses searched individuals in the GA search using the expression described in the previous section. In addition, using individuals stored in the database, a local search for the space that has not been searched is applied to use idle computing resources of enormous computing environments effectively.

4.1 Local Search

In our proposal local search, rectangles stored in the database as searched regions are expanded vertically and horizontally on a 2-dimensional plane. The rectangle (x_{min}, y_{min})-(x_{max}, y_{max}) is expanded as follows:

Rightward expansion. Set of individuals i=($x_{max}+1$, y) where $y_{min} \leq y \leq y_{max}$ should be searched and x_{max} is then updated to $x_{max}+1$.

Leftward expansion. Set of individuals i=($x_{min}-1$, y) where $y_{min} \leq y \leq y_{max}$ should be searched and x_{min} is then updated to $x_{min}-1$.

Upward expansion. Set of individuals i=(x, $y_{max}+1$) where $x_{min} \leq x \leq x_{max}$ should be searched and y_{max} is then updated to $y_{max}+1$.

Downward expansion. Set of individuals i=(x, $y_{min}-1$) where $x_{min} \leq x \leq x_{max}$ should be searched and y_{min} is then updated to $y_{min}-1$.

In one step in our proposed local search, each rectangle is expanded vertically and horizontally by 1. Fig. 5 shows an example for one step of the local search. In this figure, the area framed by the heavy line is the set of individuals that are required to search additionally.

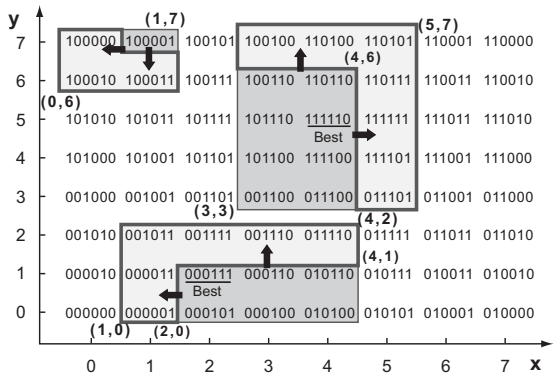

Figure 5: Example of a Local Search

The directions of longitudinal expansion and lateral expansion are determined by fitness of the best individuals on the edges of the rectangle. The direction of lateral expansion is determined as Fig. 6, where the best individual on the right edge is represented as i_a=(x_{max}, y_*), the best individual on the left edge is represented as i_b=(x_{min}, y_{**}) and the fitness of individual i is denoted F(i). The direction of longitudinal expansion is also determined in a similar way.

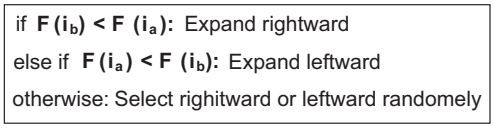

Figure 6: Direction of Expansion

The characteristics of the mapping method, which converts a multidimensional space to a 2-dimensional plane, should be considered during determination of the direction of expansion as the performance of this local search is dependent on the mapping method used. In the mapping method used in the present study, proximity on the 2-dimensional plane can correspond locally to closeness between binary strings, but this is not true globally. In expansion of a rectangle, the neighborhoods of the individuals on the edge would be searched. Therefore, it is appropriate that the best individuals on the edges are used to intensively search individuals in the direction in which good solutions were found.

Parallelization is applicable to this local search by allotting each rectangle to computation nodes. There are no strong dependencies and little communication among searches. This is expected to yield high throughput computation. Moreover, the searched space increases linearly with

increasing computing resources and an exhaustive search is guaranteed with infinite computations.

Our previous local search had the drawback that computing costs increased exponentially in accordance with the number of steps. The local search proposed here does not overweigh the search of GA as the required computing costs only increase linearly. In addition, this local search is suitable for the merge operation described in the next section to avoid overlapping of searched in the local search process in the database.

4.2 Merge Operation in the Database

Following the local search, a merge of individuals stored in the database is performed to avoid overlapping searches in the process of the local search. When there is overlap among some searched regions, they are merged to one region as follows.

When the region I_a=(x^a_{min}, y^a_{min})-(x^a_{max}, y^a_{max}) and I_b=(x^b_{min}, y^b_{min})-(x^b_{max}, y^b_{max}) satisfy $I_a \cap I_b \neq \phi$, I_b is expanded until it can include I_a, and I_a is then deleted. In this merge operation, I_b becomes $I'_b = (min(x^a_{min}, x^b_{min}), min(y^a_{min}, y^b_{min})) - (max(x^a_{max}, x^b_{max}), max(y^a_{max}, y^b_{max}))$ by searching $I'_b \cap \neg(I_b \cup I_a)$, where $min(x_1, x_2) = x_1$, $max(x_1, x_2) = x_2$ if $x_1 < x_2$. Fig. 7 shows an example for this merge.

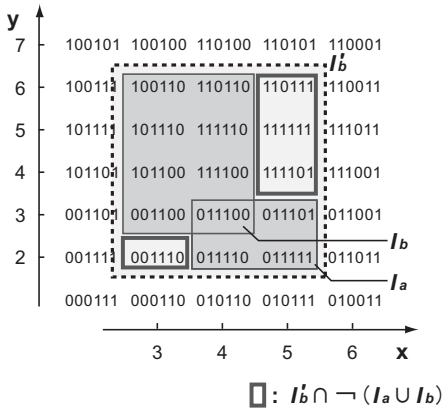

Figure 7: Example of a Merge

In this study, the merge operation was applied following one local search step because we examined our method on one local machine. When our method is implemented on large-scale computing systems, it will be necessary to synchronize the nodes that have regions that must be merged partially if there are combinations to be merged.

5 Numerical experiment

The proposed database was built into a GA, and its effectiveness was examined. Here, the searched region is expressed as proposed, and its influence on the search solution will be discussed. There are various usages for the database proposed in this research with GA, but in this experiment, a local search was conducted alternately with a GA using individuals stored in the database. To obtain a good solution at as early a stage as possible, the search was advanced based on the GA. The proposed method is outlined in Fig. 8.

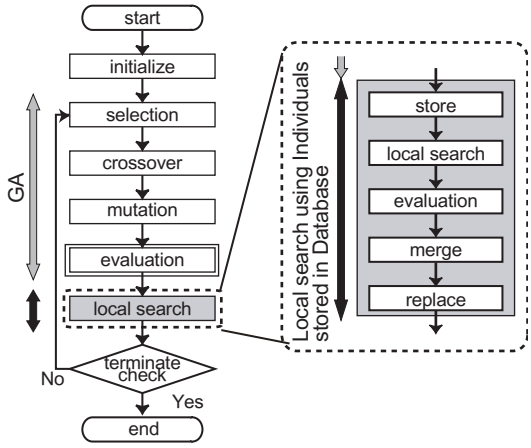

Figure 8: The Flow of a GA with the Proposed Database

First, the population in the GA is initialized. The number of individuals in the database at initial generation is 0. Next, genetic operations, such as crossover, mutation, and selection, are conducted in the population of GA, and the best individual of the population is preserved in the database. However, if this individual is already preserved in the searched region of the database, it will not be preserved again. Next, a local search operation is applied to all individuals in the database. When an individual with better fitness than the individual of interest is found, the copy of the individual will replace the worst individual in the GA population.

The database was built into the GA as described above, and its search performance was examined. The 1-max problem, which is a primitive bit test problem, the 3-deceptive problem (1) [17], which is a trap function, Rastringin function (2), Schwafel function (3), Ridge function (4), and Griewank function (5), which are continuous function test problems, were used as test problems. In the 1-max problem, the number of 1s in the chromosome is the fitness. The 1-max problem is a typical bit testing problem for GA, and the results indicated the effectiveness of the implementation of the proposed method. For the 3-deceptive problem, a search is difficult using a GA; however, the results indicated that by having the database it was possible to find an optimal solution. The effectiveness of the proposed method was also demonstrated through the testing of four kinds of continuous functions.

$$F_{3-deceptive} = \sum_{i=1}^{N} f_i \quad (1)$$

$$f_i = \begin{cases} 0.9, & u_i = 0 \\ 0.8, & u_i = 1 \\ 0.7, & u_i = 2 \\ 1.0, & u_i = 3 \end{cases}$$

$$F_{Rastrigin} = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \quad (2)$$
$$x_i \in [-5.12, 5.12)$$

$$F_{Schwefel} = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right) \quad (3)$$
$$x_i \in [-5.12, 5.12)$$

$$F_{Ridge} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2 \quad (4)$$
$$x_i \in [-64, 64)$$

$$F_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) \quad (5)$$
$$x_i \in [-512, 512)$$

5.1 Features of search of the proposed method

As the search method discussed in this research is the GA, the possibility of finding an optimal solution at a relatively early stage is high. By combining it with the proposed database, the searched region is indicated in addition to the conventional features, and even when the search is difficult using a GA, it is possible to find an optimal solution by increasing the number of calculations. In this section, a 20-bit 1max problem and a 30-bit 3-deceptive problem were used to examine the characteristics of the search conducted by a GA with a database that has the local search mechanisms built in, without limiting the calculation cost. The sizes of all search regions were 2^{20} for the former and 2^{30} for the latter. The search was finished when all solutions were evaluated. The ER model was used as the alternation in each generation. We applied a GA with uniform crossover and each couple, or parents, generated 20 children by crossover. The mutation rate was $1/L$ (L: chromosome length) and the population size was 20.

The fitness in the 1-max problem is shown in Fig. 9, in which the solid black line shows the results obtained using the proposed method (GA + LS) and the broken gray line shows the results of a conventional GA. The history of ratio of the searched region in the proposed method and the transition of the number of evaluation calculations in local search are shown in Fig. 10, in which a ratio of the search region of 1.0 indicates that all searches are complete. These results are from one trial.

As the search in the proposed method is conducted based on a GA, it has search performance equivalent to that of a conventional GA. In this problem, the optimal solution was obtained at the beginning of the search, and completion of all searches was performed to ensure that the obtained solution was optimal. In addition, by building a database into the GA, it was possible to confirm how much of the search was conducted in all of the search space to obtain the solution. Focusing on the required calculation amount for the

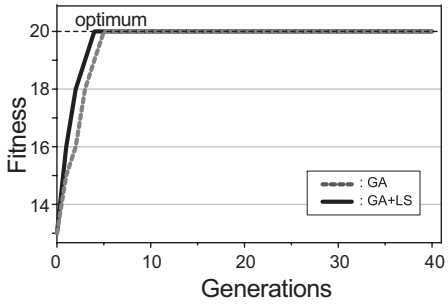

Figure 9: History of fitness in the 1-Max Problem

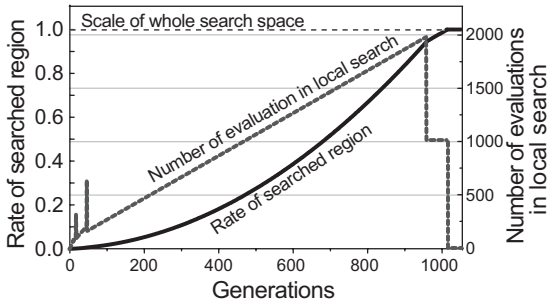

Figure 10: History of the Searched Region Rate and Number of Evaluations in the 1-Max Problem

local search at each generation for Fig. 10, it increased linearly despite being disturbed in several places due to merging, and the total searched region increased quadratically.

The transition of the fitness in the 3-deceptive function is shown in Fig. 11. The 3-deceptive problem is designed such that the population in the GA is likely to converge to a local optimum, and it is a difficult problem in which to obtain a global optimal solution using a GA. As shown in Fig. 11, the proposed method converged to the local optimum at the beginning of the search, as seen in a conventional GA, but it obtained the optimal solution by continuing the search. This was because the searched region expanded with amount of calculation resources or cost put towards the search, and the proposed method is guaranteed to obtain the optimal solution if an enormous amount of search and infinite calculation are used.

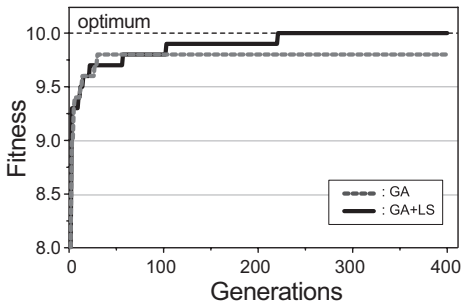

Figure 11: History of fitness in the 3-deceptive Problem

5.2 Influence of search by building the database into the GA

In this method, a large amount of evaluation calculation is necessary in the local search. When calculation resources and/or cost are limited, the GA cannot conduct a sufficient search and the performance is reduced in comparison to the conventional GA. Here, the search ability when the number of evaluation calculations is limited was examined. Four continuous functions were used to compare the search abilities of the GA and the proposed method. Both use ten-dimensional problems. The size of the population was set to 100, and the maximum number of evaluation calculations was limited to 3x10^5, 5x10^5, 1x10^6, and 2x10^6. The ER model was used for the generation alternation model in GA. The uniform crossover was used. The number of generated children in one crossover was set to 20, and the mutation rate was set to $1/L$. The results of 300 trials comparing the number at which the optimal solution was obtained are shown in Fig. 12.

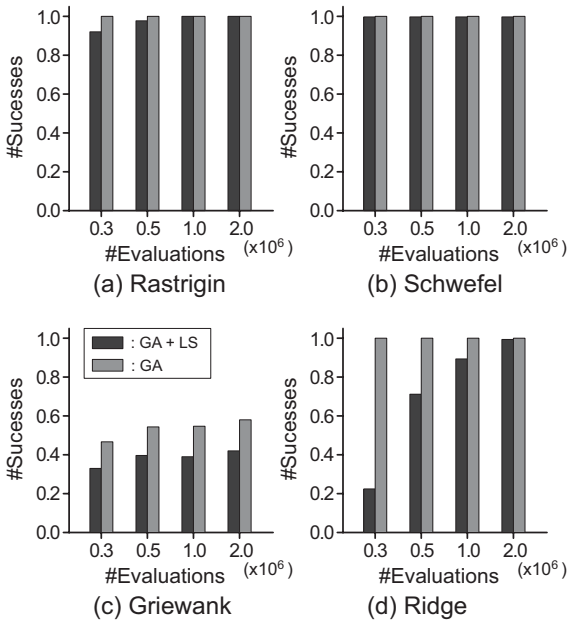

Figure 12: Number of trials at which the method obtained the optimum solution

The results shown in Fig. 12 indicate that, depending on the problem, when the calculation frequency is too small a sufficient search cannot be performed using our method and the performance is decreased. In our method, some amount of calculation is necessary to maintain the search performance of GA, and by increasing the number of evaluation calculations, the performance will improve and the search ability of GA can be almost maintained; however, decreased performance was seen in the Griewank function as compared to the conventional GA. In the proposed method, an intensive local search is conducted near the good solution, and when a better solution is obtained, it is copied to the population. This allows the population of the GA to evolve rapidly, and a good solution is expected in the early stages of the search, but the conversion is also seen earlier. This is the reason for the performance seen in the Griewank function.

One of the scales used to measure the convergence of a population is entropy[18]. Based on the frequency of the allelic gene in the loci, it is defined as a scale to measure generation efficiency through crossover and mutation, as shown in equation (6), where p_{ij} is the frequency of the allelic gene j at the locus i. L denotes the length of the chromosome and M denotes the number of allelic genes. The smaller the value of H, the gene in each locus is biased, and this indicates that the diversity of the entire population is being lost.

$$Entropy\ H \simeq \sum_{i=1}^{L}(-\sum_{i=j}^{M} p_{ij}\ \log\ p_{ij}) \qquad (6)$$

Transitions of the entropy of the proposed method (GA+LS) and GA in the Griewank function are shown in Fig. 13, which indicates the ratio to the entropy of initial population.

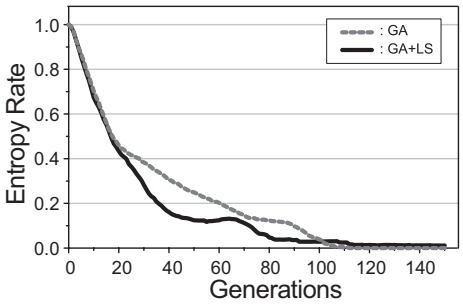

Figure 13: History of Entropy

Fig. 13 shows that the convergence of the proposed method is faster than the GA. In the proposed method, even when the search ability of the GA population is lost by convergence, it is possible to continue the local search, and the accuracy and quality of the solution will improve. To conduct an efficient search of the promising area with a small number of search points using a GA, it is necessary to recover the search ability to efficiently generate solutions through crossover or mutation.

6 Examination of Expansion of Searched Region by Applying Restart

One characteristic of the proposed method is that the searched and non-searched regions are completely separate. As repetition of the search can be avoided easily by initialization of the population in the non-searched region when the population in the GA converges, a restart strategy is suitable for our method. In addition, when the computational cost is limited, the solutions obtained by reducing the size of the population and by conducting multiple searches by restarting have been reported to be better than the solution obtained by a single search in one large population[19]. Therefore, restarting was applied to strengthen the search performance of the proposed method

and expansion of searched region under applying restart was examined.

When restarting is applied, the tabu list that stores the searched regions obtained from the previous searches is introduced in addition to the database. When the population is initialized, the searched region stored in the database is moved to the tabu list, and the database is emptied. After restarting, the good solution obtained using the GA is freshly saved in the database, and local search is conducted. No local search is conducted in the regions stored in the tabu list, and when the evaluation calculation is conducted in the GA, if the evaluated individual is in the list, repetition of the search by multiple trials and convergence to the same solution are prohibited by returning a bad value as an evaluation value.

As described above, the past search was used as the tabu list, and the search performance of the proposed method with the application of restarting was examined. Here, we will describe the improvement of search performance, and the observation that it is possible to save and expand searched regions even after restarting is applied. Restarting is applied when the population is converged. In this experiment, whether it had converged was decided based on the entropy described in the previous section, and restarting is applied when the ratio to the entropy of the initial population becomes smaller than the preset value. Although various studies have examined the timing of the restart, such as dynamic or static scheduling, this is out of scope of the present study and we will discuss this in our future work. The ratio of entropy when restarting is applied was set at multiple values of 0.01, 0.03, 0.05, 0.08, and 0.09 in this experiment. The parameters of GA were the same as in the previous section, and the ten-dimensional Griewank function was used as the testing problem.

The number of times the optimal solution was obtained is shown in Fig. 14. The transition of the ratio of the searched region, when entropy in which restarting was applied was set at 0.03, as shown in Fig. 15.

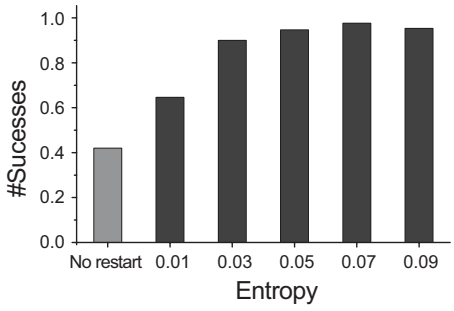

Figure 14: Number of trials where the method obtained the optimum solution

Although the performance differs depending on the timing, as shown in Fig. 14, by applying the restart, search ability constantly above a certain level can be given to the GA, and the search performance can be improved. In addition, even when the restart is applied as shown in Fig. 15, the searched region can be certainly increased and will be

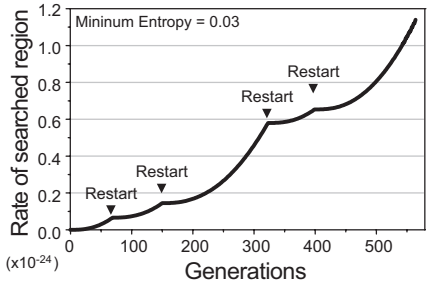

Figure 15: History of Searched region rate

expanded in accordance with the number of restart because the search is prohibited from overlapping by the tabu list. A large-scale experiment will be required to examine the effects of reference to the tabu list, because it is necessary to accumulate an enormous quantity of data in the list. This examination will be conducted in our future studies.

7 Conclusions

In this study, we proposed a GA-specific database with a local search mechanism to facilitate scalability of search performance in huge computing environments, and as an approach to achieve search capability at limited computational cost. In the database that stores the searched region to avoid repetition of the search, local searches are conducted concentrating on the non-searched region to expand the searched region. The size of the searched region can be understood quantitatively by applying the database to a GA. When the calculation resources or the computational cost is increased, the searched region is enhanced, resulting in increases in accuracy and reliability. A complete search is guaranteed when infinite calculations are performed using a huge computational resource. Our previous local search had the drawback that computing costs increased exponentially in accordance with the number of steps. We showed that the local search proposed here does not overweigh the search of GA as the required computing costs only increase linearly.

We built the proposed database into a GA, and applied the proposed method to a primitive bit testing problem and a continuous optimization problem. The results indicated that when calculation cost is unlimited, the searched region can be determined, and the solution obtained by all searches is guaranteed to be an optimal solution. Even in problems in which that GA falls easily into the local optimum, the optimal solution can be obtained by advancing the search by a local search. In addition, even when the calculation cost is limited, the results obtained were almost equivalent to those obtained by a conventional GA. Applying restart improved the search performance, and it was shown that the searched region could be certainly increased and would be expanded in accordance with the number of restart because the search was prohibited from overlapping by the tabu list.

We plan to implement the proposed method in a large computing environment and examine its effectiveness in future studies.

Bibliography

[1] H. Satoh, M. Yamamura and S. Kobayashi: Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation. Proc. of IIZUKA. pp.494-497. 1996

[2] H. Kargupta: SEARCH, polynomial complexity, and the fast messy genetic algorithm. University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Report No. 95008. 1995

[3] G. R. Harik: Linkage learning in via probabilistic modeling in the ECGA. University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Technical Report No. 99010. 1999

[4] I. Ono and S. Kobayashi: A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover. Proc. of 7th Int. Conf. on Genetic Algorithms. pp.246-253. 1997

[5] Reiko Tanese: Distributed Genetic Algorithms. Proc. 3rd International Conference on Genetic Algorithms. pp.434-439. 1989

[6] T. Jansen: On the Analysis of Dynamic Restart Strategies for Evolutionary Algorithms Proc. Parallel Problem Solving from Nature - PPSN VII, 7th International Conference. pp.33-43. 2002

[7] Alex S. Fukunaga: Restart Scheduling for Genetic Algorithms, Lecture Notes in Computer Science, vol.1498, pp.357-369. 1998

[8] Sean Luke: When Short Runs Beat Long Runs, Proceedings of the Genetic and Evolutionary Computation Conference, pp.74-80. 2001

[9] J. Maresky et al.: Selectively Destructive Restart, Proc. of Sixth International Conference on Genetic Algorithms, pp.144-150. 1995

[10] Yusuke Tanimura: Development of Master-Worker System for The Computational Grid. Information Processing Society of Japan: Computing System. vol45, No.SIG6(ACS6), pp.197-207, May 2004. in Japanese

[11] Hiroaki Imade et al.: A Grid-Oriented Genetic Algorithm for Estimating Genetic Networks by S-Systems, Proc. SICE Annual Conf. pp3317-3322, 2003

[12] Hiroaki Imade et al.: A framework of grid-oriented genetic algorithms for large-scale optimization in bioinformatics Proc. of The Congress on Evolutionary Computation in Canberra. vol.1, pp623-630, 2003

[13] H. Nakata et al.: Protain structure optimizaion using Genetic Algorithm on Jojo Journal of Information Processing Society of Japan. 2002-HPC-93, pp. 155-160, 2003. in Japanese

[14] Y. Hanada et al.: Mega Process Genetic Algorithm Using Grid MP Life Science Grid 2004 Revised Selected and Invited Papers, Lecture Notes in Computer Science, vol.3370, pp. 152-170, 2005

[15] D. Thierens, D. E. Goldberg: Elitist Recombination: an integrated selection recombination GA Proceedings of the 1st IEEE Conference on Evolutionary Computation pp.508-512. 1994

[16] Trevor D. Collins: Understanding Evolutionary Computing: A Hands on Approach, KMI-TR-48 (Knowledge Media Institute The Open University UK) September 1997

[17] Martin Pelikan et al.: BOA:The Bayesian Optimization Algorithm. IlliGAL Report No. 99003 1999

[18] N. Mori et al.: A Thermodynamical Selection Rule for the Genetic Algorithm, Proc. IEEE ICEC 95, pp.188-192, 1995

[19] David E. Goldberg Erick Cantú-Paz: Are Multiple Runs of Genetic Algorithms Better than One? Proc. Genetic and Evolutionary Computation Conference 2002, pp. 801-812, 2002.