The Eighteenth International Symposium on Artificial Life and Robotics 2013 (AROB 18th '13),
Daejeon Convention Center, Daejeon, Korea, January 30-February 1, 2013

On inhibition of premature convergence in Genetic Algorithms for mobile robot control

Satoshi Shintaku[1], and Kazushi Nakano[1,2]

[1]Dept. of Mechanical Eng. and Intelligent Systems, The University of Electro-Communications, Japan
[2]Dept. of Electronic Eng. , The University of Electro-Communications, Japan
(Tel: 81-42-443-5190, Fax: 81-42-443-5183)
(shintaku@francis.ee.uec.ac.jp)

Abstract: Methods of Evolutionary Robotics using the evolutionary computation has been applied to design of mobile robot controllers. Genetic algorithms (GAs), ones of the typical methods in the evolutionary computation, have advantages that hardly fall into local minima compared to the other optimization algorithms. However the GAs have a big problem of premature convergence that the variety of the population is reduced, so the searching ability is degraded. In this study, through analysis of a new individual generation in GAs, we propose two methods of Probabilistic Crossover and Fluctuant Mutation to inhibit the premature convergence. We apply our proposal methods to benchmark problems in optimization and to controller design of the peg pushing robot, and demonstrate the effectiveness of our proposed methods.

Keywords: Genetic Algorithms, Premature Convergence, Optimization Problem, Evolutionary Robotics

1 INTRODUCTION

Many researchers have recently focused on methods of Evolutionary Robotics (ER) using the evolutionary computation to design controllers for mobile robots [1]. In the ER, teacher signal consisting of a hypothetical input and a desired response is not required. So, the ER learns only evaluation of robot's behavior. A robot controller can be built to achieve the desired operation adapting to a variety of environments by the ER. However, the control performance depends highly on the performance of evolutionary computation itself.

Genetic algorithms (GAs), ones of the typical methods in the evolutionary computation, used for optimization have been made to mimic the process of natural evolution [2]. In the GAs, the population is built from some solutions called individuals, which are encoded binary like a genome. The GAs search the optimal solution by alternating between the evaluation of individuals and genetic operations such as crossover and mutation.

The GAs have a highly global search ability, which have a big problem of "premature convergence" . If an individual exists which has much higher fitness than others, genetic information of this individual rapidly spreads throughout the population, and the variety of the population is reduced. Because the searching ability of the GAs depends on the variety of the population, the searching ability is degraded by the occurrence of premature convergence, and the population has a possibility of converging to local solutions.

Although there are the methods to increase the number of individuals so as to maintain the variety of the population, an increase in the number of individuals causes to an increase in the computational effort, so there is a risk that a robot controller cannot be implemented in real time. In robot controller design using the ER, the premature convergence tends to occur because of the difficulties in the fitness function design and hardware restrictions such as robot sensors. Therefore GAs approaches that enable to inhibit the premature convergence are required.

In this study, we discuss about the changes in the variety of population by crossover and mutation in GAs using uniform crossover. We divide the convergence of individuals into two parts; one is "learning convergence" that comes from the difference between the fitness of individuals, and the other is "nature convergence" that occurs independently of the fitness of individuals. Next we propose two methods of Probabilistic Crossover (PC) and Fluctuant Mutation (FM). The PC is a crossover method for determining the probability distribution of the allele, and to generate individuals of new generation according to the probability distribution, so the PC is a crossover method for putting forward learning convergence. The FM is a method for tetermining the mutation rate so as to complement the decrease of variance based on the natural convergence from the variances of the individuals after crossover.

We apply our proposal methods to benchmark problems in optimization and to controller design of the peg pushing robot, and demonstrate the effectiveness of the methods in terms of the quality and robustness of solution and the convergence speed of learning.

2 RELATED WORKS

The GAs introduced by J. H. Holland in the 1970s are optimization algorithms that mimic the evolution of life. The

GAs have the following features: first, the GAs are iterative calculation methods, which update concurrently multiple solution candidates, secondly, the GAs use unique operations called "crossover" that encode candidate solutions to binary data called "genotype" and generate new candidate solutions by combining genotype, finally the GAs are flexible algorithms that can be applied to a wide variety of problems.

There are two methods for analysis of the GAs, one is Schema theory and the other is for analysis using Markov chain. The Schema theory determines the change of a subset of genes (Schema) that are components of each individual in Simple Genetic Algorithms (SGAs) using one-point crossover and mutation [3]. The Schema theory shows that the SGAs can increase exponentially the schema which has partially favorable information and get globally favorable solutions. However the Schema theory cannot give the convergence to the optimal solution.

The analysis of the Markov chain calculates the probability of discovery of the optimal solution and shows the convergence to the optimal solution in the GAs [4]. From the Markov chain analysis, the probability of discovery of the globally optimal solution converges to "1" after calculating in an infinite time if keeping the storage of the best individual that has been found so far and having the positive mutation rate. However the assumption is independent of the main algorithms of the GAs such as crossover and selection. It is synonymous with that the probability of discovery of the optimal solution converges to 1 in random search.

The Schema theory and the Markov chain analysis show that the GAs get the globally optimal solution after calculating in an infinite time, on the contrary, these cannot show the quality of the solution obtained in a realistic time. So we do not get exactly how to design many parameters used in the GAs from the Schema theory and the Markov chain analysis.

Meanwhile, the GAs have improved its searching ability of the optimal solution by absorbing the advantage of the other optimization algorithms and creating a new method.

Thermodynamical Genetic Algorithms (TDGAs) select the individual of the next generation based on Free Energy Minimization at Simulated Annealing [5]. The TDGAs can control explicitly the variety of individual by setting temperature schedule and avoid the premature convergence. On the other hand, the TDGAs have the following problems: the TDGAs require more computational time than the SGAs, and the TDGAs causes the TDGAs convergence that loses the variety of the individual due to approximate calculation of the entropy.

Distributed Genetic Algorithms (DGAs) are ones of the GAs methods that make some islands that the population is divided into, and the crossover and evaluation are performed [6]. The DGAs provide the immigration that lets individuals move between the islands once in several generations. Emigrated individuals are the candidates of the optimal solution in other islands, so they have a higher fitness compared to mutated individuals and are hardly lost by selection. Emigrated individuals are candidates of the solution optimized the different situation, so they can contribute to the maintenance of variety compared to mutated individuals changed only a small part of the genome. As a result, the DGAs can obtain a higher fitness than GAs that learn in a single population. But at the same time, because the number of individuals per island is reduced, the environment of each island makes it easy to converge the local minimum. Therefore, the improvement of the searching ability in DGAs has never been theoretically proven yet.

Diploid Genetic Algorithms (Diploid GAs) have two genotypes and represent a phenotype through a new type called "agency-type" [7]. Because the Diploid GAs have the locus that does not appear in the phenotype, does not affect the fitness and carry over low fitness genome to the next generation, it is possible to maintain the variety of individuals.

These methods are said to get a higher searching ability than the GAs. However the parameters that should be set up are increased, for example, temperature scheduling in the TDGAs, the number of islands and the timing of immigration in the DGAs, and the methods for converting genotype to phenotype in the Diploid GAs. In order to search the optimal solution efficiently, it is necessary to determine these parameters appropriately.

3 PROBABILISTIC CROSSOVER AND FLUCTUANT MUTATION

3.1 Analyses of Genetic Algorithms

In this section, we discuss about the crossover and the mutation of the GAs from the viewpoint of probability theory. First, we derive an expression for the probability distribution of the allele of a new individual generated by the crossover. The analyzed GAs consist of the roulette selection and the uniform crossover.

Selecting an individual $\boldsymbol{s_i}$ as the parent, the probability $p(\boldsymbol{s_i})$ that the i-th individual becomes a factor for determining a new individual's locus is described as

$$p(\boldsymbol{s_i}) = \frac{f(\boldsymbol{s_i})}{\sum_{i=1}^{N} f(\boldsymbol{s_i}).} \tag{1}$$

where N is the number of individuals, and $f(\boldsymbol{s_i})$ is the fitness for a given $\boldsymbol{s_i}$.

The probability $p(x'_j = 1)$ that the j-th locus of the generated individual is equal to 1 is described as

$$
\begin{aligned}
p(x'_j = 1) &= \sum_{i=0}^{N-1} p(\boldsymbol{s_i}|x_{ij} = 1) \\
&= \frac{\sum_{i=0}^{N-1} f(\boldsymbol{s_i}) x_{ij}}{\sum_{i=0}^{N-1} f(\boldsymbol{s_i})} \\
&= \frac{p(x_{ij} = 1)F}{1 + p(x_{ij} = 1)(F - 1),} \qquad (2) \\
F &= \frac{ave(f(\boldsymbol{s_i})|_{x_{ij}=1})}{ave(f(\boldsymbol{s_i})|_{x_{ij}=0}).} \qquad (3)
\end{aligned}
$$

where x_{ij} is the j-th locus of the i-th individual of parent generation, and $ave(f(\boldsymbol{s_i})|_{x_{ij}=1})$ is the average value of $f(\boldsymbol{s_i})$ if x_{ij} is equal to 1.

Figure 1 shows some examples of the above equation.

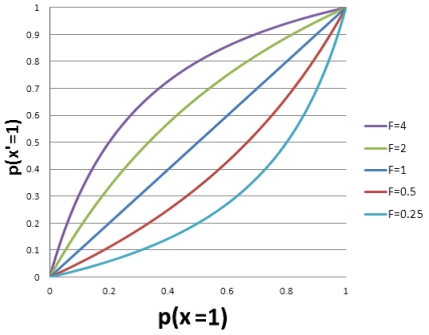

Fig. 1. Change of $p(x)$ from crossover in GAs

Next, we discuss about a new population named S to generate the N-th individual from the probability distribution $p(x')$. Here, $R[S|x' = 1]$ is the rate that the locus is equal to 1 in S, and $E[\mathrm{var}[S]]$ represents the expectation of the variance of S. These are represented as

$$
\begin{aligned}
R[S|x' = 1] &= p(x' = 1), \qquad (4) \\
E[\mathrm{var}[S]] &= \frac{N-1}{N} p(x' = 1)p(x' = 0). \qquad (5)
\end{aligned}
$$

As shown by the above equation, $E[\mathrm{var}[S]]$ is smaller than the variance of $p(x')$ because of sample variance.

Finally, when the population S is mutated with the probability P_m, we show $E[\mathrm{var}[S']]$ as follows: the probability that locus x^* representing mutated locus x of S is equal to 1 is calculated by

$$
p(x^* = 1) = p(x = 1) + P_m (p(x = 0) - p(x = 1)). \qquad (6)
$$

Therefore, $E[\mathrm{var}[S']]$ is described by

$$
\begin{aligned}
E[\mathrm{var}[S']] &= (P_m - P_m^2) \left(\frac{N-1}{N} - 4\mathrm{var}[x] \right) \\
&+ E[\mathrm{var}[S]]. \qquad (7)
\end{aligned}
$$

From the above equation, we find that the mutation acts in a direction to increase the variance, and that this action works strongly in the proportion to the variance in prior mutation.

3.2 Consideration from analyses of GAs

Figure 1 shows that the crossover changes the probability of the locus in the direction of increasing the fitness. As a result, the individuals converge to the genome having a higher fitness as the generation repeated. This is defined as "learning convergence".

Furthermore, Eq. (5) shows that the generation of the individuals reduces the variance independently of the fitness. It means that the individuals converge independent of the fitness with the generation. This is defined as "natural convergence".

Figure 1 shows that the difference is small between the probability distribution of the allele of generated individuals and the rate of the genome in the parent generation. In other words, $p(x' = 1) - p(x = 1)$ is very small. Thereby, in the case of converging in a wrong direction, even if F in Eq. (3) is determined to correct the wrong convergence, there is a risk that the natural convergence cancels a modification of learning by the learning convergence. This causes the premature convergence of the GAs.

In our study, we propose a probabilistic crossover method for determining the probability distribution of the allele of generated individuals in crossover, and also do a fluctuant mutation for changing the mutation rate according to the variance of the locus. We intend to solve this premature convergence problem.

3.3 Probabilistic Crossover and Fluctuant Mutation

The probabilistic crossover of our first proposing method is one of the crossover methods for calculating the probability distribution of the allele of generated individuals:

$$
p(x' = 1) = \begin{cases} F * p(x = 1) \, (F < 1) \\ 1 - \frac{(1-p(x=1))}{F} \, (F \geq 1) \, , \end{cases} \qquad (8)
$$

and generate individuals according to this probability distribution instead of selecting parent individuals and crossing over the genome.

Equation (8) is designed so as to be a tangent line to the point $p(x = 1) = 0$ or $p(x = 1) = 1$ in (2). Figure 2 shows some examples of (8).

In $F > 1$ we design $p(x)$ to increase an increment of $p(x)$ at around $p(x = 1) = 0$, the learning convergence is likely to occur even in the neighborhood of $p(x = 1) = 1$ or 0 while the premature convergence is less likely to occur.

The fluctuant mutation is our second method for estimating the reduction of the variance by the natural convergence and for determining the mutation rate to complement the lost variance.

Since the effects of the natural convergence are different in each locus, the mutation rate uses different values in each locus. Moreover, the mutation handles only the individuals

generated by the crossover and does not effect the individuals selected by the elite selection.

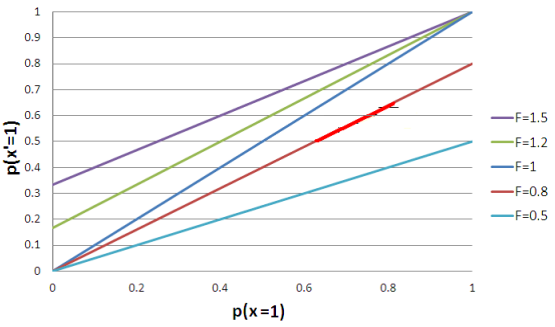

Fig. 2. Change of $p(x)$ from probabilistic crossover

Thus, the fluctuant mutation must complement the natural convergence that occurs in the entire population with the mutation only for the individuals generated by the crossover. So, the mutation rate $p_m(\mathrm{var}[x'])$ in the fluctuant mutation is determined by

$$p_m(\mathrm{var}[x']) = \frac{1}{P_c}\left(\frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N-1}\frac{\mathrm{var}[x']}{\frac{N-1}{N} - 4\mathrm{var}[x']}}\right). \quad (9)$$

In the above equation, if $\mathrm{var}[x']$ equals to 0, $p_m(\mathrm{var}[x'])$ will be 0. This does not fulfill the condition that the mutation rate is a positive value in the Markov chain analysis, and the convergence to the optimal solution is not guaranteed. Hence, by extending Eq. (9) to Eq. (10), the fluctuant mutation avoids this problem.

$$p_m(\mathrm{var}[x']) = \begin{cases} \frac{1}{P_c}\left(\frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N-1}\frac{\mathrm{var}[x']}{\frac{N-1}{N} - 4\mathrm{var}[x']}}\right) & (\mathrm{var}[x'] \geq \epsilon) \\ P_m & (\mathrm{var}[x'] < \epsilon) \end{cases} \quad (10)$$

The use of the fluctuant mutation inhibits the convergence of the variance of the allele by the natural convergence as well as the premature convergence.

4 SIMULATION RESULTS

4.1 Application to optimization problem

To investigate the performance of the proposed methods, each optimization algorithm is evaluated through the typical benchmark problems in optimization. The optimization algorithms to be compared are the GAs using the uniform crossover and the constant mutation rate as the conventional method. The GAs using the probabilistic crossover (PCGAs), the GAs using the fluctuant mutation (FMGAs), and the GAs using both the probabilistic crossover and the fluctuant mutation as the proposed methods. We examine the learning processes for the cases that the number of elite individuals is given as 30, and the number of original individuals is given as 50, 100 and 200.

4.1.1 Rastrigin function

The aim of each optimization algorithm is to find the minimum value of Rastrigin function:

$$F(x) = 10N + \sum_{i=0}^{N}\left(x_i^2 - 10\cos(2\pi x_i)\right) \quad (11)$$

$$-5 \leq x_i < 5. \quad (12)$$

The Rastrigin function has a global minimum at $x = 0$ where $F(x) = 0$, and it has the suboptimal solutions in a reticular pattern near the global minimum. The fitness function is defined by

$$fitness = 1 - \frac{F(x)}{(20 + 5^2)N}. \quad (13)$$

Table 1 summarizes the maximum fitness values for the 100-dimensional Rastrigin function at the 1000-th generation in each learning method. These values show the average of 100 trials, and within the parenthesis shows their standard deviation.

Table 1. Fitness for Rastrigin function

Individuals	50	100	200
GAs	0.919(0.006)	0.941(0.005)	0.945(0.006)
PCGAs	0.957(0.004)	0.962(0.004)	0.963(0.004)
FMGAs	0.939(0.008)	0.951(0.005)	0.952(0.004)
PCFMGAs	0.778(0.063)	0.964(0.004)	0.968(0.003)

Figures 3 and 4 show the transitions of the maximum fitness values and their variances in 100 individuals.

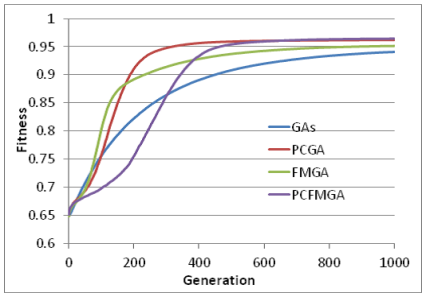

Fig. 3. Transition of fitness for Rastrigin function

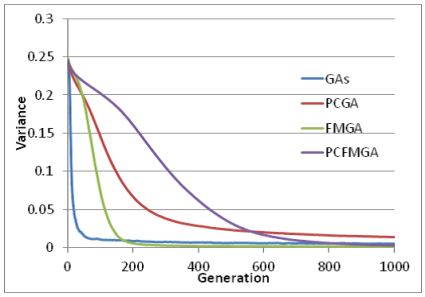

Fig. 4. Transition of variance of fitness for Rastrigin function

4.1.2 Schwefel function

The aim of each optimization algorithm is to find the minimum value of the Schwefel function:

$$F(x) = -\sum_{i=0}^{N}\left(x_i \sin(\sqrt{|x_i|})\right) \quad (14)$$

$$-500 \leq x_i < 500. \quad (15)$$

The Schwefel function has a global minimum at $x = 420$ where $F(x) = -418.98N$, and it does not have the suboptimal solutions near the global minimum. The fitness function is defined by

$$fitness = 1 - (F(x) + 418.98N)/840N. \quad (16)$$

Table 2 summarizes the maximum fitness values for the 100-dimensional Schwefel function at the 1000-th generation in each learning method. These values show the average of 100 trials, and within the parenthesis shows their standard deviation.

Table 2. Fitness for Schwefel function

Individual	50	100	200
GAs	0.83(0.011)	0.88(0.012)	0.89(0.011)
PCGAs	0.95(0.006)	0.93(0.005)	0.95(0.006)
FMGAs	0.91(0.009)	0.92(0.008)	0.92(0.008)
PCFMGAs	0.95(0.006)	0.96(0.006)	0.97(0.005)

4.1.3 Rosenbrock function

The aim of each optimization algorithm is to find the minimum value of the Rosenbrock function:

$$F(x) = -\sum_{i=0}^{N-1}\left(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\right) \quad (17)$$

$$-2 \leq x_i < 2. \quad (18)$$

The Rosenbrock function has a global minimum at $x = 1$ where $F(x) = 0$, and it has a high dependence between the variables. The fitness function is defined by

$$fitness = 1 - \sqrt{F(x)/3609/(N-1)}. \quad (19)$$

Table 3 summarizes the maximum fitness values for the 100-dimensional Rosenbrock function at the 1000-th generation in each learning method. These values show the average of 100 trials, and within the parenthesis show their standard deviation.

4.2 Controller design for peg pushing robot

We simulate the problem of peg pushing robot control as an application task. The control task is to push the peg toward the goal by a two-wheel robot. The feedforward neural network (NN) having three layers is used as the robot controller. The inputs of robot controller are the relative coordinates of the goal and peg, and the outputs are the rotational speed of the wheel. The number of the hidden layer of the NN is 10.

Table 3. Fitness for Rosenbrock function

Individual	50	100	200
GAs	0.941(0.005)	0.956(0.005)	0.961(0.005)
PCGAs	0.959(0.005)	0.963(0.005)	0.964(0.004)
FMGAs	0.943(0.005)	0.954(0.005)	0.956(0.005)
PCFMGAs	0.946(0.014)	0.961(0.004)	0.965(0.004)

In this simulation, each optimization algorithm optimizes the synapse of the NN. The fitness function is defined by

$$fitness = \exp\left(-\frac{dist(\mathrm{goal}, \mathrm{peg_{end}})}{dist(\mathrm{goal}, \mathrm{peg_{start}})}\right) \quad (20)$$

where, $dist(\mathrm{goal}, \mathrm{peg_{start}})$ is the initial distance between the peg and goal, and $dist(\mathrm{goal}, \mathrm{peg_{end}})$ is the final distance between the peg and goal.

We evaluate the fitness values of controllers obtained from 10 different initial positions of the peg.

The optimization algorithms used in this simulation are the GAs, TDGAs, DGAs and Diploid GAs as the conventional methods and the PCFMGAs as proposed method. The number of individuals is 100 in each optimization algorithms, and the number of elite selection is 30 other than the TDGAs, and the mutation rate is 0.1% in the conventional methods, and P_m is 0.01% in the FMGAs. In the DGAs, the number of the island is 5, and the migration interval is 20. The temperature schedule of TDGAs is defined by

$$T(t) = \exp\left(-\frac{t}{25}\right). \quad (21)$$

Table 4 summarizes the maximum fitness values and success rate that the maximum fitness value is greater than 0.95 at the 1000-th generation in each learning method. The maximum fitness value shows the average of 100 trials, and within the parenthesis shows their standard deviation.

Table 4. Maximum fitness and success rate for robot controller

Alghrithm	Maximum Fitness	Success Rate
GAs	0.912(0.102)	0.48
TDGAs	0.675(0.160)	0.04
DGAs	0.889(0.079)	0.20
Diploid GAs	0.758(0.186)	0.30
PCFMGAs	0.952(0.076)	0.72

Figure 5 shows the transition of the maximum fitness values.

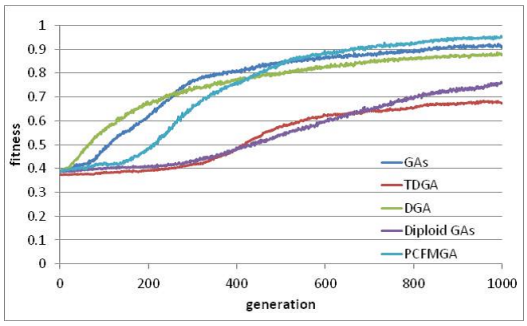

Fig. 5. Transition of fitness of peg pushing robot

5 CONSIDERATION OF PROPOSED METHODS

5.1 Learning method for benchmark problems

Tables 1 - 3 show that the PCGAs gain higher fitness values than the GAs in all benchmark problems. In contrast, the FMGAs cannot gain higher fitness values than the GAs in the Rosenbrock function, and the fitness of the FMGAs tends to be lower than that of the PCGAs in all the benchmark problem. While the PCFMGAs gain the highest fitness values among the other algorithms in the case of many individuals, the fitness values of the PCFMGAs are lower than the other algorithms in the case of a few individuals.

Figure 4 proves the convergence of the individuals of the GAs at around 100-th generation. For this reason, the learning convergence does not work well in the GAs, and only the mutation. Exponentially increasing of the fitness value in Fig. 3 shows that searching the optimal solution become heuristic by the mutation.

Figures 3 and 4 show that the PCGAs and FMGAs slowly reduce the variances of the individuals, and the learning processes according to the convergence of the individuals. The PCGAs maintain the variances to some extent even if the learning progresses at the long generations. On the contrary, the FMGAs lose the variances after the generation. Therefore, the PCGAs gain a higher fitness than the FMGAs in the final result of learning. Thus, a proper setting of P_m in the FMGAs can gain a higher fitness.

The PCFMGAs make the convergence of the individuals very slow, and aquire the high global search ability. As a result, the PCFMGAs gain the highest fitness values among all the algorithms after the long generation. However the PCFMGAs are not suitable in the case of a limited number of generations because of slow convergence.

5.2 Learning method for robot controller design

Figure 5 shows that the TDGAs, DGAs and Diploid GAs do not gain higher fitnesses than the GAs in design of the mobile robot controller. Table 5 shows that the success rate is low while the DGAs gain a high fitness. The fact demonstrates that the DGAs do not have the local search ability while it has the global search ability.

Moreover Fig. 5 shows the potential that the Diploid GAs gain s higher fitness at more advanced generation. However, it requires longer computational time than the other algorithms. Hence, each algorithm is not suitable for optimization of the robot controller design.

On the other hand, the PCFMGAs of the proposed methods gain the highest fitness and success rate among all the optimization algorithms. The number of generations required in the PCFMGAs nearly equals to that required in the GAs and DGAs. Hence, the PCFMGAs are not suitable for optimization of the robot controllerdesign.

6 CONCLUSION

To inhibit the premature convergence which is a problem in the mobile robot controller design, we proposed the new crossover method and the method for determining the mutation rate, through the analysis of a new individual generation in GAs. We applied our proposal methods to the benchmark problems in optimization and to the controller design for the peg pushing robot, and demonstrated the effectiveness of our proposed methods.

REFERENCES

[1] Toshiyuki Kondo: Evolutionary design and behavior analysis of neuromodulatoryneural networks for mobile robots control: Applied Soft Computing, Volume 7, Issue 1, pp.189-202,2007

[2] J. H. Holland: Adaptation in Natural and Artificial Systems: University of Michigan Press, 1975

[3] David E. Goldberg: A Practical Schema Theorem for Genetic Algorithms Design and Tuning: IlliGAL Report, No. 2001017, January, 2001

[4] A. E. Eiben,E. H. L. Aarts and K. M. Van Hee: Global convergence of genetic algorithms: A markov chain analysis: Lecture Notes in Computer Science Volume 496, pp.3-12, 1991

[5] Naoki Mori, Junji Yoshida, Hisashi Tamaki, Hajime Kita and Yoshikazu Nishikawa: A thermodynamical selection rule for the genetic algorithm : Evolutionary Computation, 1995, IEEE International Conference on

[6] Xiong Shengwu: A distributed genetic algorithm to TSP: Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on, pp.1827-1830, volume 3, 2002

[7] Shengxiang Yang: On the Design of Diploid Genetic Algorithms for Problem Optimization in Dynamic Environments: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pp.1362-1369 , 2006