The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th '10),
B-Con Plaza, Beppu,Oita, Japan, February 4-6, 2010

A novel coding method for genetic algorithms based on redundant binary number

A. Murayama and A. Kanasugi

Graduate School of Engineering, Tokyo Denki University
2-2 Kanda-Nishiki-cho, Chiyoda-ku, Tokyo, 101-8457 Japan
(Tel: 03-5280-3705; Fax: 03-5280-3565)
(Email address: 09kme47@ms.dendai.ac.jp, kanasugi@eee.dendai.ac.jp)

Abstract: This paper proposes a novel GA which switches the expression of solution from redundant binary number to usual binary number. Furthermore, a GA which switches the expression from Gray code to usual binary number is compared. Gray code is a binary numeral system where two successive values differ in only one bit. In this paper, comparison of performance among five GA (binary number, redundant binary number, Gray code, switching from redundant binary number to binary number, switching from Gray code to binary number) were illustrated. The performance were evaluated by solving some equations. It is confirmed that the proposed GA was effective for improvement of error rate.

Keywords: Genetic Algorithm, Redundant binary number, Gray code**.**

I. INTRODUCTION

Genetic algorithm (GA) is one of optimization algorithm that is based on an idea for evolution of life [1]. GA can solve various problems such as combination optimization problem, machine learning, and so on.

This paper proposes a novel GA based on redundant binary number, though general GA expresses chromosomes in binary number. The performance comparisons were simulated by solving some equations. It was confirmed that the proposed GA was effective for improvement of error rate.

II. PROPOSED METHOD

In this paper, GA is based on redundant binary number. The advantage with redundant binary number is variety of expression for optimized solution. From this advantage, we can expect that improvement in searching speed and decrease of error rate. The redundant binary number uses values '0', '1' and '-1'. We assume that '0' sets "00" or "11", '1' sets "01", '-1' sets "10". The procedure of GA based on redundant binary number is almost the same as conventional GA. However, when GA based on redundant binary number evaluates individual, it is necessary to decode from redundant binary number to binary number. In this paper, as a decoding method, we separate chromosomes into odd number bit and even number bit. Then we subtract even number bit from odd number bit.

This paper proposes a novel GA based on redundant binary number [2]. The proposed GA can avoid suboptimal by variety of expression pattern with redundant binary number, in the first-half generation. Then the GA can search in detail with binary number in the second-half generation.

In addition, number of bits for the solution expression doubles in redundant binary number compared with usual binary number. Therefore, number of individuals can be doubled by switching the construction from redundant binary number to binary number in the second-half generation.

Furthermore, a GA which is realized by switching the expression from Gray code to usual binary number is compared.

III. EXPERIMENTS

Comparison among five GA (binary number, redundant binary number, Gray code, switching from redundant binary number to binary number, switching from Gray code to binary number) were simulated. The performance of each GA was evaluated by solving following three functions [3].

1) Rosenbrock function

$$f_1(x) = \sum_{i=1}^{9} \left[100 \times \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right] \quad (1)$$

2) Weighted sphere function

$$f_2(x) = \sum_{i=1}^{10} \left(10000 - x_i^2 \right) \quad (2)$$

3) Schwefel function

$$f_3(x) = -\sum_{i=1}^{5} x_i \sin\sqrt{|x_1|} \qquad (3)$$

$f_1(x)$ has dependence among variables and it is unimodal function.

$f_2(x)$ doesn't have dependence among variables and it is unimodal function.

$f_3(x)$ doesn't have dependence among variables and it is multimodal function. The parameters of GA are summarized in table 1. However, number of population for redundant binary number becomes the half of the parameter.

Table 1. GA parameters

Function	(1)	(2)	(3)
Generation	300	200	500
Population	128		
Selection	Roulette		
Crossover	One Point		
Crossover Rate	0.99		
Mutation Rate	0.01		

All GA program are implemented in C language. The error rates of three functions are shown in figure 1, 2 and 3, respectively. Each result is the average of 1000 times of simulation. In these figures, 'B' means binary number, 'G' means Gray code, "G2" means switching from Gray code to binary number, 'R' means redundant binary number, and "R2" means switching from redundant binary number to binary number.

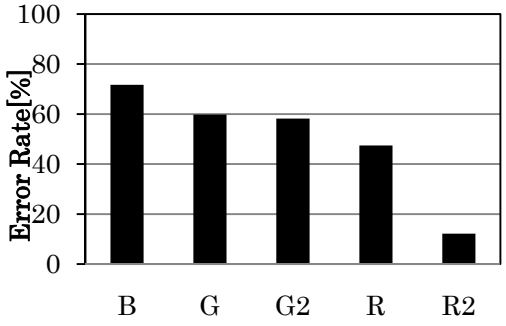

Fig.1. Error rate of $f_1(x)$

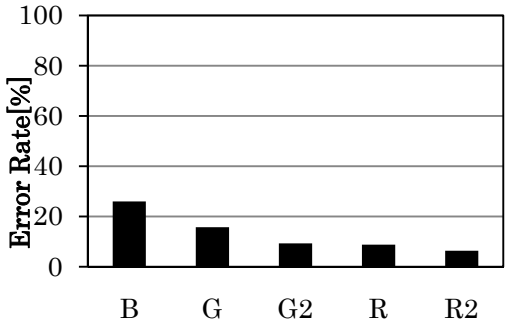

Fig.2. Error rate of $f_2(x)$

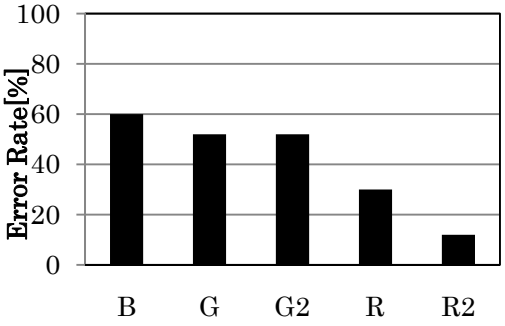

Fig.3. Error rate of $f_3(x)$

In figure 1, solutions tend to converge in the suboptimal solution by strong dependence among variables. However, GA with redundancy binary number tend to avoid suboptimal solution by variety of chromosomes expression and gets the optimal solution. In figure 2, though error rate didn't have much difference, GA with redundant binary number showed good results. Similarly, in figure 3, the proposed GA gave good results.

From these results, it was confirmed that proposed GA was effective for improvement of error rate.

IV. CONCLUSION

In this paper, we proposed a novel coding method for genetic algorithms based on redundant binary number. As the novel code increases the variety of chromosomes expression, the proposed GA is effective to escape from local optimum solution.

It was confirmed that proposed GA was effective for improvement of error rate by three test function.

REFERENCES

[1] L. Davis ed. (1991), Handbook of Genetic Algorithms, Van Nostrand Reinhold

[2] A. Tsukahara and A. Kanasugi (2009), Genetic Algorithm with Dynamic Variable Number of Individuals and Accuracy International Journal of Control, Automation, and Systems, vol. 7, no.1: 1-6

[3] X. Yao, Y. Liu, and G. M. Lin (1999), Evolutionary programming made faster, IEEE Trans. Evol. Comput., vol. 3, no. 2: 82–102