A Coevolutionary Approach To Adapt The Genotype-Phenotype Map In Genetic Algorithms

Hajime Murao, Hisashi Tamaki, and Shinzo Kitamura
{murao,tamaki,kitamura}@al.cs.kobe-u.ac.jp
Department of Computer and Systems Engineering, Faculty of Engineering, Kobe University
Kobe 657-8501, JAPAN

Abstract - This paper introduces a coevolutionary approach to genetic algorithms (GAs) for exploring not only wihtin a part of the solution space defined by the genotype-phenotype map but also the map itself. In canonical GAs with the fixed map, how large area of the solution space can be covered by possible genomes and consequently how better solutions can be found by a GA rely on how well the genotype-phenotype map is designed, but it is difficult for designers of the algorithms to designe the map without *a-priori* knowledge of the solution space. In the proposed algorithm, the genotype-phenotype map is improved adaptively during the searching process for solution candidates. It is applied to 3-bit deceptive problems as a kind of typical combinatorial optimization problems, which are well-known by that the diffculty against GAs can be controlled by the genotype-phenotype map, and shows fairly good performance beside a conventional GA.

I. INTRODUCTION

Genetic algorithms (GAs) can be applied to the task finding feasible solutions for optimization problems where a solution candidate is encoded into a computer-editable description as a genome. Solution candidates are progressively improved by applying simulated genetic operations like mutation and recombination onto a population of genomes. There are really large number of applications of GAs[8], [7].

However, in general, it cannot be assured that a GA can explore enough large area of the solution space defined by possible solution candidates for the given problem to find reasonable solutions. It depends on so-called the genotype-phenotype map, an decoding manner from genomes to the corresponding solution candidates as phenotypes, but it is difficult to obtain *a-priori* knowledge of the problem to determine the suitable map. Therefore, GA is usually executed with a single correspondence between genomes and their phenotypes namely the map is constant. Consequently, even if using the GA-based approaches, experienced human designer of GAs are still required to explore the genotype-phenotype map itself for finding the better one.

This paper introduces a coevolutionary approach to relax the difficulty. It consists of two populations of genomes where one is used for exploring inside the part of solution space to find the solution and another for exploring the genotype-phenotype map. This enables to explore not only within a part of the solution space defined by the genotype-phenotype map but also the map itself. The original idea of the proposed method comes from a viewpoint of supporting the engineering design as stated above, but it is also an useful enhancement for GAs themselves to relax native difficulty existed in conventional GAs to determine the genotype-phenotype map without *a-priori* knowledge of the solution space.

In the following, we firstly give brief overview for related studies and then introduce the proposed method with discussions about its expectable performance from this point of view in connection with the schema theorem. It is applied to 3-bit deceptive problems introduced by Goldberg which are well-known by that the difficulty against GAs can be controlled by the genotype-phenotype map. Finally, we conclude.

II. RELATED STUDIES

There have been studies on so-called fitness landscape in which the influence of genetic operations over the shape of the explorable area of GAs are discussed[6], [10]. Ranges of parameters of genetic operations are proposed but can be applied for limited problems only. Consequently, adaptive ways to adjust parameters according to the problem are becoming more popular in these days[2], [9], [11]. Genetic operations determine the topology of the genetic space and are closely related to the performance of GAs too.

Concerning about the genotype-phenotype map, a well-known characteristic of the string type genome is the schema theorem, which gives an aspect for the evolutionary process that long, high-order schemata will have a less survival probability. Therefore, it is thought better

0-7803-7282-4/02/$10.00 ©2002 IEEE

to place genes related each other at closer on genomes. To realize it without *a-priori* knowledge of the problem, methods operating a string of 2-tuples of a gene and its position on a genome are proposed[5], [3].

There is another interesting work proposing a method to find not a genotype-phenotype mapping directly but better schemata as building-blocks for relaxing the difficulty of finding better genotype-phenotype mapping[4], where a coevolutionary approach is utilized. It is closely related interesting approach to our method.

There are also articles trying to theoretically analyze the effect of the genotype-phenotype map on the performance of GAs from the point of biological view[1], [13]. This will help to explain results of our approach.

III. THE COEVOLUTIONARY APPROACH

A. Seed, Rule, Genotype and Phenotype

The proposed method consists of two populations of genomes, where one consists of sequences of values and another indicates positions of corresponding values in a genotype of a solution candidate. The former corresponds to genes and the latter corresponds to loci. A genome of a solution candidate is generated by rearranging a sequence of genes by using corresponding loci information. In the following, to avoid confusion, we refer a sequence of genes as a **seed**, a loci description as a **rule**, and a genome of a solution candidate regenerated from a seed and a rule as just a **genome**. Their roles are not restricted but intuitively a population of rules determine a part of the solution space and a population of seeds explores inside it. In this sense, modifying rules by applying genetic operations onto them corresponds to explore the genotype-phenotype map itself.

A seed $\boldsymbol{s}$ can be defined according to *a-priori* knowledge of the problem if you have. For instance, you can define $\boldsymbol{s}$ as $\boldsymbol{s} \in R \times R \cdots R$ if you know the real-coded vector type genome is enough to describe and explore any solution candidates. We basically assume a binary string genome in the following and consequently it is enough to think about a binary string seed $\boldsymbol{s} \in \{0,1\} \times \{0,1\} \cdots \{0,1\} = \{0,1\}^l$ where l is the number of genes in a genome, but the method itself is not restricted to binary string genomes. A rule $\boldsymbol{r}$ is defined as an permutation of numbers indicating positions of corresponding bits in a genome, *i.e.* a rule $\boldsymbol{r}$ can be defined as $\boldsymbol{r} \in L \times L \cdots L = L^l$ where $L = \{1, 2, ..., l\}$ if the duplication of elements is allowded.

Let S and R as a set of seeds and rules respectively. We indicate a genome generated by applying the j-th rule $\boldsymbol{r}_j$ of R onto the i-th seed $\boldsymbol{s}_i$ of S as $\boldsymbol{x}_{ij}$. We can define any combination manner of $\boldsymbol{s}_i$ and $\boldsymbol{r}_j$ to generate $\boldsymbol{x}_{ij}$, and apply *a-priori* knowledge of the problem here. Two simple cases for the seed $\boldsymbol{s}_i = (s_i(1), s_i(2), ..., s_i(l))$ and the rule $\boldsymbol{r}_j = (r_j(1), r_j(2), ..., r_j(l))$ are

$$x_{ij}(k) = s_i(r_j(k)), \tag{1}$$

and

$$x_{ij}(r_j(k)) = s_i(k). \tag{2}$$

We would like to note here that bits in a genome are potentially left undefined when using the combination manner Eq.2 with rules for which the duplication of elements is allowded. In such case, special treatments for the undefined bits are necessary. The most simple way is to introduce "default value".

In both cases, the definition length of sub-strings of the seed $\boldsymbol{s}_i$ and the rule $\boldsymbol{r}_j$ on what schemata of the genome $\boldsymbol{x}_{ij}$ is mapped are related to k. That is, schemata of a genome are mapped onto same long sub-strings of a corresponding rule using the combination manner in Eq.1 while the length of sub-strings of a seed are independent to the length of schemata. In this case, the length of sub-strings of a seed can become short as enough.

On the other hand, by using the combination manner in Eq.2, the length of sub-strings of a rule is equal to that of a corresponding seed but they both are independent to the length of schemata of the genome. In this case, the length of sub-strings of both a seed and a rule can become short as enough. From this point of view, Eq.2 is more flexible and is thought to be superior to Eq.1 but it should be investigated more carefully.

B. Evaluation of Seed and Rule

A fitness value of a genome $\boldsymbol{x}_{ij}$ can be calculated by a combinatorial function $f(\cdot)$ of mapping from a genome to a solution candidate and mapping from a solution candidate to a fitness value. Some kind of fitness values for rules and seeds can be thought but we here introduce the maximum fitness value of genomes generated from a seed as a fitness value of the seed and also the maximum fitness value of genomes generated by a rule as a fitness value of the rule as follows

$$f_s(\boldsymbol{s}_i) = \max_{\boldsymbol{r}_j \in R} f(\boldsymbol{x}_{ij}), \tag{3}$$

$$f_r(\boldsymbol{r}_j) = \max_{\boldsymbol{s}_i \in S} f(\boldsymbol{x}_{ij}). \tag{4}$$

C. Algorithm

The procedure of the proposed algorithm is as follows:

1. **Initialization**: Initialize both sets S and R of seeds and rules by randomly generated strings.

TABLE I
FITNESS VALUES OF 3-BIT STRINGS

bit-string $\boldsymbol{y}$	$e(\boldsymbol{y})$		bit-string $\boldsymbol{y}$	$e(\boldsymbol{y})$	
	P1	P2		P1	P2
(0,0,0)	28	0	(1,0,0)	14	26
(0,0,1)	26	14	(1,0,1)	0	28
(0,1,0)	22	30	(1,1,0)	0	0
(0,1,1)	0	0	(1,1,1)	30	22

2. **Evaluation of genomes**: Generate possible genomes by using seeds and rules in the current sets of S and R, and evaluate them.
3. **Evaluation of seeds and rules**: Calculate fitness values of seeds and rules as in Eq.3 and Eq.4.
4. **Genetic operations**: Apply genetic operations onto seeds and rules to generate new sets S' and R' of seeds and rules.
5. We call a procedure from 2 to 4 as a generation. Repeat generations until conditions to terminate are satisfied.
6. Output a solution candidate corresponding to a genome with the maximum fitness value over generations as a result.

IV. COMPUTER SIMULATIONS

A. 3-bit Deceptive Problems

The deceptive problem is an optimization problem in which large and smooth basins for not the global optimum but local optima are defined so as to mislead the search to the sub-optimal solutions. It is done by violating the static building-block hypothesis, *i.e.* when genomes are 2-bit strings, the four potential building-blocks of $\{0*, *0, 1*, *1\}$ are schemata, and the static building-block hypothesis is violated if fitness values are defined as $f(0*) > f(1*)$ or $f(*0) > f(*1)$ if the global optimum is at point 11.

The 3-bit deceptive problems are formulated as a problem finding a combination of N 3-bit strings $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_N\}$ which maximizes the fitness value given by $\sum_{i=1}^{N} e(\boldsymbol{y}_i)$, where $e(\boldsymbol{y}_i)$ is a fitness value of 3-bit strings $\boldsymbol{y}_i = (y_i(1), y_i(2), y_i(3))$ $(y_i(j) \in \{0,1\}\ \forall j)$ defined so as to violate the static building-block hypothesis.

We prepare two $N = 50$ problems P1 and P2 where the optimal 3-bit strings are $\boldsymbol{y}_i = (1,1,1)$ $(\forall i = 1, ..., N)$ and $\boldsymbol{y}_i = (0,1,0)$ $(\forall i = 1, ..., N)$ respectively. Fitness values for every 3-bit strings are given by Table I.

B. Conventional GA

For the sake of comparison, a conventional GA with single population of genomes are applied to the deceptive problems, where three decoding methods of "loose coding", "tight coding" and "random coding" are used. A genome $\boldsymbol{x} = (x(1), x(2), ..., x(l))$ where length $l = 3N$ and $x(i) \in \{0,1\}$ for all $i = 1, 2, ..., l$ is decoded into a solution candidate $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_N\}$ as follows:

loose coding the solution candidate is composed of 3 bits at intervals of N bits.

$$y_k(m) = x(k + (m-1)N), \quad m = 1, 2, 3 \tag{5}$$

tight coding the solution candidate is composed of a series of 3 bits.

$$y_k(m) = x(3(k-1) + m), \quad m = 1, 2, 3 \tag{6}$$

random coding the solution candidate is composed of randomly chosen 3 bits.

$$y_k(m) = x(\text{random}). \tag{7}$$

The main characteristic of the coding is the definition length which denotes how far bits belonging the same string are located on the genotype. The definition length of "loose coding" is N, that of "tight coding" is 1, and the average of the definition length of "random coding" is $(l+1)/(3+1)$. This is closely related to the difficulty of the problem, *i.e.* the larger definition length means the difficult problem. In this case, "loose coding" completely decomposes 3-bits strings into a genome and makes the problems the most difficult while "tight coding" make them easy.

We apply tournament selection with the size=2, the one-point crossover, and the bit-flip mutasion as the genetic operations. We also apply so-called elitest strategy where a genome corresponding to the best solution candidate in a generation is forced to remain in the population of the next generation. A fitness value of a solution candidate is defined so as to normalized in a range $[0, 1]$ as,

$$\frac{1}{30N} \sum_{i=1}^{N} e(\boldsymbol{y}_i), \tag{8}$$

and parameters are summarized in Table II. We have tried different values for the crossover rate and the mutation rate and the best so far are used.

Figure 1 and Fig. 2 are the convergence curves of the maximum fitness values averaged over 30 trials and standard deviations are also shown. It is shown that the optimal solutions with fitness value 1.0 are found in both problems by using "tight coding" and the worse solutions by "loose coding".

TABLE II
PARAMETERS FOR THE CONVENTIONAL GA

Population size	10,000
Crossover rate	60 % of population
Mutation rate	0.67 % per gene

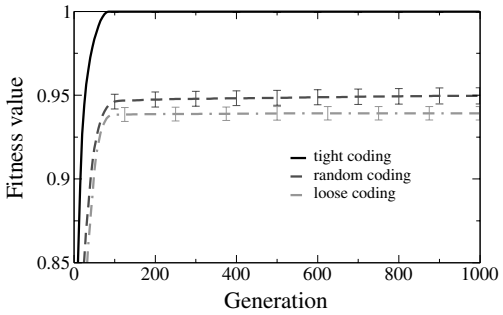

Fig. 1. The maximum fitness values averaged over 30 trials for the conventional GA applied to the problem P1

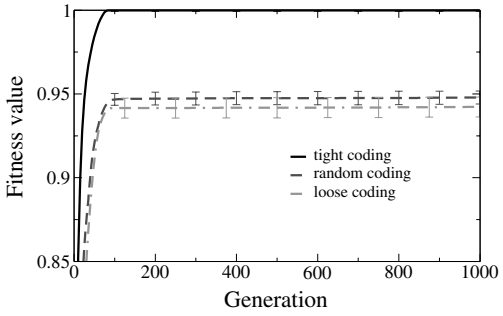

Fig. 2. The maximum fitness values averaged over 30 trials for the conventional GA applied to the problem P2

C. Application of the Proposed Method

The propose method is applied to the problems. Let the size of the set S be 100, for which the length of a seed $\boldsymbol{s}_i \in S$ is $l = 3 \times N = 150$ and the k-th element of a seed is as $s_i(k) \in \{0, 1\}$, and let the size of the set R be also 100, for which the duplication of elements in a rule is not allowded, *i.e.* the k-th element of a rule $\boldsymbol{r}_j \in R$ is as $r_j(k) \in \{1, 2, ..., l\}$ where $r_j(a) \neq r_j(b)$ for all $a \neq b$ $(a, b \in \{1, 2, ..., l\})$. Consequently, 10,000 genomes, the same number of genomes with the case of conventional GA, are evaluated in every generations.

We try both of decoding manners described in Eq. (1) and Eq. (2) to generate genomes. Every genome is mapped to a solution candidate according to so-called "loose coding" manner as described in Eq.5. A fitness value of a solution candidate is defined so as to normalized in a range $[0, 1]$ as described in Eq.8. We apply the tournament selection with the size=2, the one-point crossover, the bit-flip mutation as the genetic operations for the set S of seeds, and the tournament selectoin with the size=2, PMX[12], and bit-swapping between different two positions as the mutation for the set R of rules. We also apply so-called elitest strategy where a seed and a rule corresponding to the best solution candidate in a generation are forced to remain in the sets S and R of the next generation. Other parameters are summarized in Table III.

TABLE III
PARAMETERS FOR THE PROPOSED METHOD

Population size of seed	100
Population size of rule	100
# of evaluated genomes	10,000
Crossover rate	60 % of population
Mutation rate	0.67 % per seed/rule

Figure 3 and Fig. 4 show time courses of the maximum fitness values averaged over 30 trials, in which "type 1" indicates the decoding manner in Eq. 1 and "type 2" for Eq.2. Despite of using "loose coding", the proposed method outperforms the conventional GA with "random coding" in both problems.

D. Discussion

The proposed method shows quite good performance for the problem P2. This means that the coevolutionary search proposed here is working well. At first sight, the result might be strange since the problem P1 appears easier than P2, because once a seed $\boldsymbol{s}$ with $s(i) = 1$ $\forall i$ is obtained, any rule can be applied to generate the optimal solution and there becomes no need to search for a rule in P1. However, if coevolutionary search works well and rules can contribute enough, the possible number of seeds with fifty "1"s out of 150 bits, which can be translated to the optimal solution in P2, is ${}_{150}C_{50}$. It is quite larger than ${}_{150}C_{150} = 1$, the possible number of seeds can be translated to the optimal solution in P1.

We then observe of the proportion of "1"s in the elitest genome given by Eq.9.

$$\text{Proportion of "1"} = \frac{1}{l} \sum_{k=1}^{l} x(k), \tag{9}$$

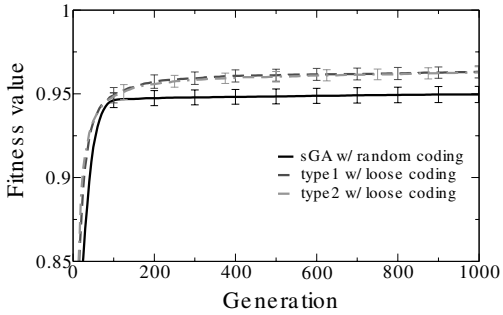

Fig. 3. Time courses of the maximum fitness values averaged over 30 trials for the problem P1

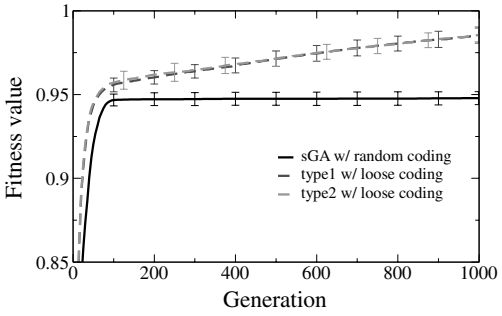

Fig. 4. Time courses of the maximum fitness values averaged over 30 trials for the problem P2

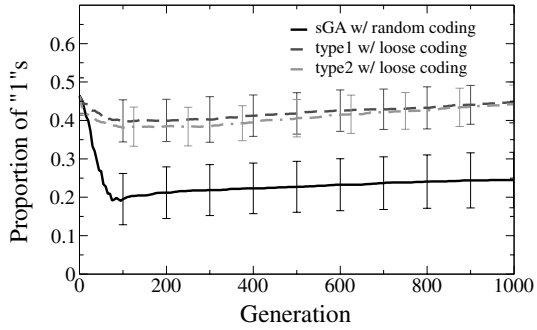

Fig. 5. Proportion of "1"s to the bit-length for the problem P1.

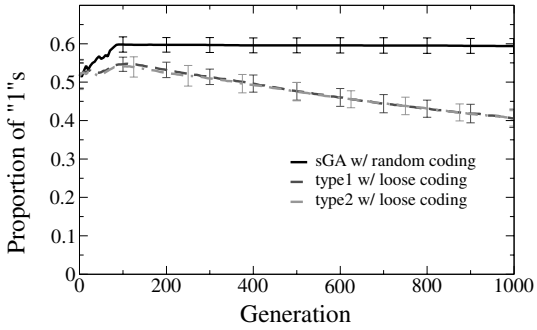

Fig. 6. Proportion of "1"s to the bit-length for the problem P2.

where $x(k)$ is the value of the k-th bit of the target genome $\boldsymbol{x}$. Figure 5 and Fig. 6 show the proportions averaged over 30 trials against problems P1 and P2 respectively. Since the optimal solution of the problem P2 is $\boldsymbol{y} = (0, 1, 0)$, the proportions should be distributed around $1/3 = 0.333$ after sufficient number of generations in Fig. 6. Actually, the proposed method leads the proportions to around 0.333 while the simple genetic algorithm mislead to the sub-optimal solution of $\boldsymbol{y} = (1, 0, 1)$ with the proportions around $2/3 = 0.666$. In Fig. 5 where 1.0 corresponds to the optimal solution $\boldsymbol{y} = (1, 1, 1)$, the behavior of the proposed method is not so radical than that in Fig. 6 but it is still better than the conventional one with keeping the proportions higher.

To know how rules work well, we investigate results for the problem P2 from the viewpoint of the distribution of the optimal string $\boldsymbol{y} = (0, 1, 0)$ and the sub-optimal string $\boldsymbol{y} = (1, 0, 1)$ within seeds and genomes. Figure 7 shows the rate of the strings in the populations of seeds and Fig. 8 shows the rate in generated genomes. The rate of the optimal string contained in seeds not dramatically but slightly increases in Fig. 7 but that contained in genomes monotonically increases with generations. These results show that the rearranging seeds by rules works very well.

The remaining question is which combination manner is better than another. To investigate this, we introduce the definition length d_1 and d_2 for the codings of Eq. 1 and Eq. 2 respectively. Which are calculated as follows,

$$d_1 = \frac{1}{(3-1)N} \sum_{j=1}^{N} (\max_{m=1,2,3} \{r_j(k + (m-1)N)\} - \min\{r_j(k + (m-1)N)\}), \quad (10)$$

$$d_2 = \frac{1}{(3-1)N} \sum_{j=1}^{N} (\max\{r_j^{-1}(k + (m-1)N)\} - \min\{r_j^{-1}(k + (m-1)N)\}), \quad (11)$$

where $r_i^{-1}(x) = j$ for $r_i(j) = x$. They are normalized within a range $[0, 1]$.

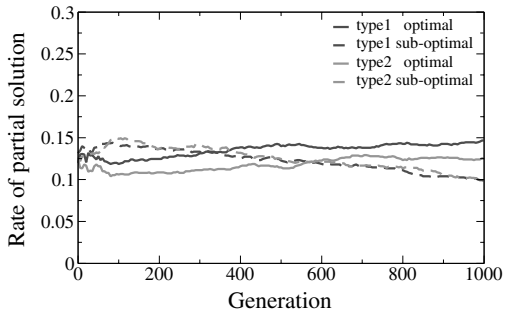

Fig. 7. The rate of the optimal and the sub-optimal strings within seeds.

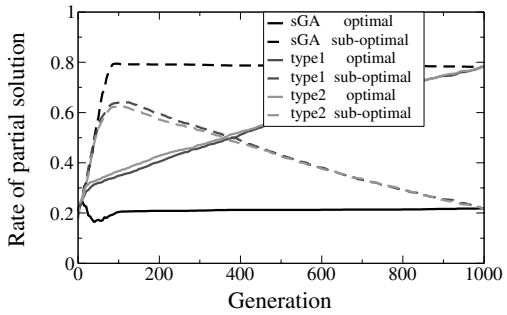

Fig. 8. The rate of the optimal and the sub-optimal strings within seeds and genomes.

Figure 9 shows the definition length of the best genome for the problem P2 averaged over 30 trials. It is observed that the definition length is fluctuated during a trial and between trials around 0.75 which denotes easier than "loose coding" but more difficult than "tight coding". There are no significant trend with generations and also no differences between the two codings.

V. Conclusion

In this paper, We proposed a coevolutionary GA to evolve not only genomes but also the genotype-phenotype map. The proposed method was applied to the 3-bit deceptive problems and outperformed a simple GA.

Acknowledgements

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), 12450160, 2001.

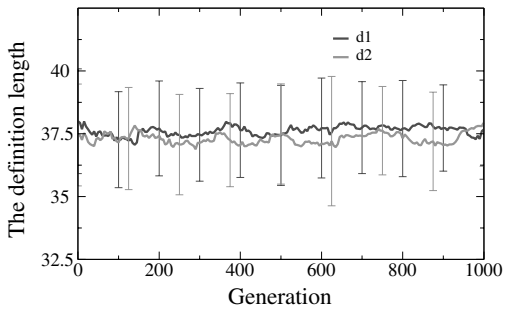

Fig. 9. The averaged definition length of the best genomes for the problem P2.

References

[1] L. Altenberg. Genome growth and the evolution of the genotype-phenotype map. In W. Banzhaf and F. H. Eeckman, editors, *Evolution and Biocomputation: Computational Models of Evolution*, pages 205–259. Springer, 1995.

[2] T. Bäck. Optimal mutation rates in genetic search. In *Proc. of the FIfth ICGA*, pages 2–8, 1993.

[3] D. Goldberg, K. Deb, and B. Korb. Don't worry, be messy. In *Proc. of ICGA'91*, pages 24–30, 1991.

[4] H. Handa, N. Baba, and O. Katai. Genetic algorithm involving coevolution mechanism to search for effective genetic information. In *Proc. of the 1997 IEEE Int. Conf. on Evolutionary Computation*, pages 709–714, 1997.

[5] J. Holland. *Adaptation In Natural And Artificial Systems*. MIT Press, 1992.

[6] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *J. theor. Biol.*, 128:11–45, 1987.

[7] S. Kitamura, Y. Kakuda, H. Murao, J. Gotoh, and M. Koyabu. A design method as inverse problems and application of emergent computations. *J. of SICE*, 36(1):90–97, 2000. In Japanese.

[8] S. Kitamura, Y. Kakuda, and H. Tamaki. An approach to the emergent design theory and applications. *J. of Artificial Life and Robotics*, 3:86–89, 1999.

[9] C. Maley. The coevolution of mutation rates. In F. Moran, A. Moreno, J. Merelo, and P. Chacon, editors, *Advances in Artificial Life*, pages 234–245. 1995.

[10] K. Mathias and D. Whitley. Genetic operators, the fitness landscape and the traveling salesman problem. In *Proc. of PPSN*, volume 2, pages 219–228, 1992.

[11] W. M. Sears. Adaptive crossover in evolutionary algorithms, 1995.

[12] T. Starkweather, D. Whitly, C. Whitly, and K. Mathial. A comparison of genetic sequencing operators. In *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, pages 69–76, 1991.

[13] G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. In *Evolution*, volume 50, pages 967–976, 1996.