Chemical Genetic Algorithms: Enhancing Evolvability of GAs via Genotype-Phenotype Mapping

Hidefumi Sawai[1], Hideaki Suzuki[2], and Wojciech Piaseczny[2]

[1] National Institute of Information and Communications Technology
588-2 Iwaoka, Nishi-ku, Kobe 651-2274, Japan
sawai@nict.go.jp
[2] ATR Network Informatics Laboratories
2-2-2 Hikari-dai, Keihanna Science City, Kyoto 619-0288, Japan
{hsuzuki, wojtek}@atr.jp

Abstract. We propose the concept of a *chemical genetic algorithm* (CGA), in which several types of molecules (information units) react with each other in a cell. Not only the information in DNA but also that of the smaller molecules responsible for the transcription and translation of DNA into amino acids is changed adaptively during evolution. This mechanism optimizes the fundamental mapping from binary substrings in DNA (genotypes) to real values for a parameter set (phenotypes). Through the struggle between cells containing a DNA unit and small molecular units, the codes (DNA) and the interpreter (the small molecular units) co-evolve, and a specific output function, from which a cell's fitness is evaluated, is optimized. To demonstrate the effectiveness of the CGA, it is applied to some problems including a set of deceptive problems and benchmark problems such as Shekel's foxholes and a generalized Langermann's function. To ascertain the validity of the genotype-phenotype mapping by the CGA, some analytical experiments were conducted while observing the basin size of a global optimum solution in the binary genotypic space. The results show that the CGA effectively broadens the basin size, making it easier to find a path to a global optimum solution, while enhancing the GA's evolvability during evolution.

1 Introduction

From the design standpoint of an evolutionary system, the translation from genotype to phenotype plays a critical role. The translation specifies the system's genotype-to-phenotype mapping, the fitness landscape in the genotype space, and ultimately, its evolvability, meaning its ability to evolve to reach advantageous solutions. Evolvability is one of the most fundamental properties of an evolutionary system, and its enhancement is essential to the design of a 'good' evolutionary system. In most studies of artificial evolutionary systems, however, the translation relation specifying the fitness landscape is determined by a human designer *prior to* an experimental evolutionary run. If we could evolutionarily optimize the translation based on the initial setting that we prepare, we might be able to evolve a more effective genotype-to-phenotype mapping. This paper's primary topic is a method for enabling such evolutionary optimization of the translation. We can expect improvement in the performance or possibility of artificial evolutionary systems from such co-evolution between the genotype-phenotype translation and the genotypes.

When we consider biological systems, on the other hand, the translation of the genetic information in DNA is not fixed in advance but changes through evolution. The fundamental mapping from DNA codes (codons) to functional units (amino acids) is specified by a set of molecules called aminoacyl-tRNAs, which are created by translating codes in DNA [1]. The final fitness of a chromosome is calculated not only from the DNA information but also from the chemical reactions conducted by the entire molecular set in a cell. Hence, in a biological system, not only the DNA but also other smaller molecules are responsible for the phenotypic molecules used by selection. Although a mutation is basically exerted only on the DNA, modifications of chromosomes are reflected to the other internal molecules through transcription or translation. As a result, during biological evolution, DNA and the translation molecules (tRNAs and aminoacyl-tRNAs) can be optimized simultaneously.

Borrowing this mechanism, this paper proposes a new method for a genetic algorithm (which we refer to as a chemical genetic algorithm, CGA) that enables the coevolution between DNA codes and their translation and enhances evolvability [4][5]. We prepare a population of artificial cells that include four different types of molecular units: a DNA string, tRNA strings, amino acid units (Aminos), and aminoacyl-tRNA units (aa-tRNAs). A population of cells having this structure is evolved by using operations for selection, DNA mutation, DNA crossover, molecular exchange, and chemical reaction, while the cell's fitness is evaluated from the target function, which is calculated from the specific output amino acid values. To assess the effectiveness of the CGA, we apply it to functional optimization problems that are hard to solve by a simple GA (SGA), and analyze the evolution of CGA focusing on the enhancement of evolvability.

2 The Model

The CGA uses a population of cells, each of which contains a set of artificial molecules. The molecular set is made up of a DNA string, tRNA strings, amino acid units (Aminos), and aminoacyl-tRNA units (aa-tRNAs)(Fig. 1). The DNA unit and tRNAs are represented by binary strings, the Aminos are represented by real values normalized between zero and one, and the aa-tRNAs are represented by combinations of a binary string and a real value between zero and one. All of the binary strings are codons or indices, where an index is prepared in imitation of the identifier sequence of a real tRNA which specifies correspondence between a codon and an amino acid.

2.1 Parameters in CGA

The parameters used in the model are summarized below.

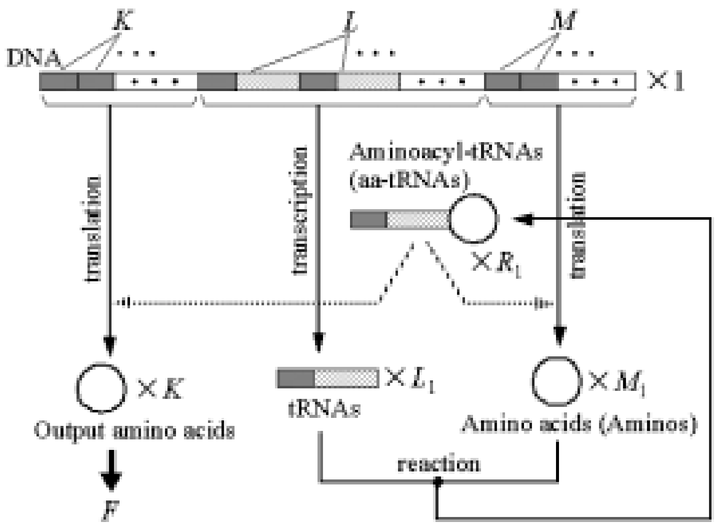

Fig. 1. A cellular structure used in the CGA. The dark hatched rectangles are codons described with short binary strings, the bright hatched rectangles are indexes described with binary strings, and the circles are amino acids described with real numbers. The dashed translation from DNA to Aminos does not happen with the CGA(NF).

J_1 : Codon length (number of bits in a codon),
J_2 : Index length (number of bits in an index),
K : Number of output amino acids for the target protein, or the number of dimensions for the target function F,
L : Number of tRNA units in the DNA,
M : Number of codons for Aminos in the DNA,
I : Total length of (number of bits in) the DNA $= KJ_1+L(J_1+J_2)+MJ_1$,
L_1 : size limit of the tRNA pool (maximum number of tRNAs in a cell),
M_1 : size limit of the Amino pool (initial/maximum number of Aminos in a cell),
R_1 : size limit of the aa-tRNA pool (maximum number of aa-tRNAs in a cell),
R_2 : Reaction rate of tRNAs and Aminos, or the number of aa-tRNAs created by the tRNA-Amino reaction per cell per generation,
p_{m} : Mutation rate (probability of flipping of DNA bits) per bit per generation,
p_{c} : Crossover rate (occurrence probability of one-point crossover between a DNA pair) per cell pair per generation,
N : Population size (the number of cells in the population),
β : Exponent for the target function F,

a,*b* : Fitness coefficient for linear scaling,
c : Fitness coefficient for exponential scaling.

3 Chemical Genetic Algorithm

As the CGA procedure is described in detail elsewhere [4][5], here we simply describe its generational operation cycle as follows (see Fig. 2).

1. **[Initialization]** Prepares a population of cells with a random DNA string and amino acids, then operate the generation cycle until a termination criterion is satisfied.
2. **[Chemical reaction]** During the first several generations, create new tRNA and aa-tRNA molecules, making their pool sizes grow.
3. **[Selection]** Calculate the cells' fitness values from the output amino acids and conduct roulette-wheel selection by using the fitness values.
4. **[DNA mutation]** Execute the conventional mutation (bit flipping) operation on the DNA strings of the cells.
5. **[DNA crossover & molecular exchange]** Mate all cells to make $N/2$ pairs. For each pair, execute the conventional crossover (exchange of DNA substrings) operation and a molecular exchange operation.
6. Examine the population, and terminate if the criterion is satisfied. Otherwise, go to Step 2.

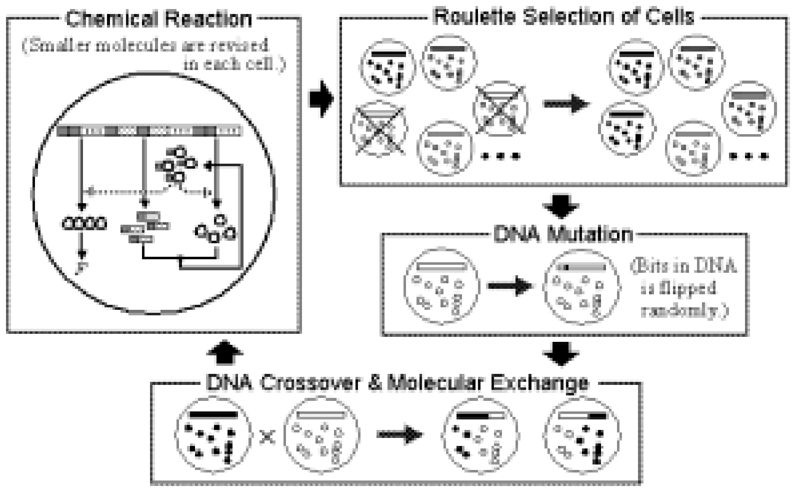
Chemical Reaction
(Smaller molecules are revised in each cell.)
F
Roulette Selection of Cells
DNA Mutation
(Bits in DNA is flipped randomly.)
DNA Crossover & Molecular Exchange

Fig. 2. Generation cycle of the CGA.

4 Experiments

To verify the effectiveness of the CGA, we applied it to different functional optimization problems that are difficult to solve with the SGA, and we compared the results to those obtained with the SGA and the PfGA [3].

4.1 Deceptive Problems

The first problem relates to a K-dimensional deceptive function. A deceptive problem is a kind of problem in which the total size of the basins for local optimum solutions is much larger than the basin size of the global optimum solution and GAs are very often deceived into climbing local optimum solutions. The deceptive function $F(x)$ is defined as

$$F(x) = \Big(\frac{1}{K}\sum_{k=0}^{K-1} f_k(x)\Big)^{\beta}, \tag{1}$$

with a non-linearity factor β and the final fitness calculated from $fitness = a + bF(x)$ for linear scaling or $fitness = \exp(cF(x))$ for exponential scaling by using the constant numbers a, b, and c. We defined three types of deceptive problems.

A complex deceptive problem (Type III), in which the global optimum is located at $x = \alpha_k$, where α_k is a unique random number between 0 and 1 depending on the dimension k $(k = 0, 1, \cdots, K-1)$, can be formulated as

$$f_k(x) = \begin{cases} -\frac{x}{\alpha_k} + \frac{4}{5} & \text{if } 0 \le x \le \frac{4}{5}\alpha_k \\ \frac{5x}{\alpha_k} - 4 & \text{if } \frac{4}{5}\alpha_k < x \le \alpha_k \\ \frac{5(x-\alpha_k)}{\alpha_k - 1} + 1 & \text{if } \alpha_k < x \le \frac{1+4\alpha_k}{5} \\ \frac{x-1}{1-\alpha_k} + \frac{4}{5} & \text{if } \frac{1+4\alpha_k}{5} < x \le 1. \end{cases} \tag{2}$$

The two other types of deceptive problems (Types I & II) are special cases of the complex deceptive problem, with $\alpha_k = 1$ (Type I), or $\alpha_k = 0$ or 1 at random (Type II) for each dimension k $(k = 0, 1, \cdots, K-1)$ [4][5].

For all three types of $f_k(x)$, the region with local optima is $5^K - 1$ times larger than the region with a global optimum in the K-dimensional space. The number of local optima is $2^K - 1$ for Type I and Type II deceptive problems and $3^K - 1$ for Type III.

We performed experiments with the three types of deceptive problems in the five- and ten- dimensional cases. According to the results of some preliminary experiments, we set the following experimental conditions: linear scaling ($a = 0$, $b = 1$) for the SGA, and exponential scaling ($c = 20$) for the CGA.

Table 1 summarizes the set of parameter values used for the CGA (see Sec. 2.1 for the definitions). Table 2 compares the results of the SGA and two different CGAs for five and ten dimensions. The computational cost of the CGA was about ten times larger than that of the SGA. The CGA outperformed the SGA by far for deceptive problems of Types I and II, whereas the CGA and the SGA*

Table 1. Parameter set for the CGA

J_1	J_2	K	L	M	L_1	M_1	R_1	R_2	I	Pm	Pc	N	β
6	8	2/5/10	64	128	320	1280	320	64	1676/1694/1724	0.005	0.7	256	5

(the SGA with 10,000 generations) showed comparable performance in solving the Type III deceptive problem. We consider these results to derive from the fact that the CGA adaptively optimizes the coding relation for a given target function.

Table 2. Success ratios for the SGA, CGAs, and PfGA [3] in solving deceptive problems

GA	SGA	SGA*	CGA(WF)	CGA(NF)	PfGA [3]
J_1	6	6	6	6	20
scaling	linear	linear	exp.	exp.	none
T_{max}	1,000	10,000	1,000	1,000	1,000**
F_{th}	0.8	0.8	0.8	0.8	0.8
Type(K=5)	2%	9%	100%	100%	100%
I (K=10)	0%	3%	100%	100%	100%
Type(K=5)	12%	20%	100%	100%	100%
II(K=10)	0%	5%	100%	100%	100%
Type(K=5)	47%	85%	95%	100%	100%
III(K=10)	14%	79%	43%	94%	100%

For the results shown in Table 2, we calculated a success ratio from 100 runs with different random number sequences. For example, 2% means that two runs out of 100 succeeded in finding the global optimum solution before reaching the maximum number of generations, $T_{\max}$. The CGA(WF) is an original model with translation feedback by aa-tRNA, while the CGA(NF) is a modified model with no feedback. **The PfGA performed 300 runs and evaluated the same number of individuals, $N \cdot T_{\max} = 256{,}000$, as the SGA and CGAs for each run.

4.2 Benchmark Problems

The next problems that we considered were two benchmark problems, Shekel's foxholes and a generalized Langermann's function [2]. These problems have correlations between multiple input variables (variable-inseparable) and global solutions that cannot be found by considering each input variable independently. We normalized each function so that the maximum functional value would be approximately equal to one. The original forms of the functions are given by

$$F(x) = -\sum_{j=1}^{m} \frac{1}{\sum_{k=1}^{K}(x_k - a_{jk})^2 + c_j}, m = 30 \qquad (3)$$

for Shekel's foxholes, and

$$F(x) = -\sum_{j=1}^{m} c_j \exp\left[-\frac{1}{\pi}\sum_{k=1}^{K}(x_k - a_{jk})^2\right] \cos\left[\pi\sum_{k=1}^{K}(x_k - a_{jk})^2\right], m = 5 \quad (4)$$

for the generalized Langermann's function, where a_{jk} and c_j are constant numbers fixed in advance. We specified linear scaling with $a = 0$ and $b = -1/10$ for the SGA and exponential scaling with $c = 20$ for the CGA in the case of Shekel's foxholes. For the generalized Langermann's function, we specified linear scaling with $a = 0.6$ and $b = -1/1.75$ for the SGA and exponential scaling with $c = 20$ for the CGA. These were the best conditions that we obtained through preliminary experiments. Using different random number sequences, we conducted 100 different runs with the SGAs and CGAs for the two-, five-, and ten-dimensional ($K = 2$, 5, 10) benchmark problems.

Table 3 compares the results that we obtained for each problem with the different algorithms. As the number of dimensions, K, increases, the difficulty of the benchmark problems increases geometrically. Accordingly, with $K = 2$, all of the algorithms could almost always find the global solution, as judged by the criterion, $F(x) > 0.9$, but with $K = 10$, most of the algorithms failed to find a global optimum solution. With $K = 5$ for Shekel's foxholes, the CGAs, and especially CGA(NF) (i.e., the CGA with no translation feedback by aa-tRNA), significantly outperformed the SGA. Furthermore, the CGA showed performance comparable to that of the PfGA [3], which is one of the most powerful genetic algorithms developed thus far.

Table 3. Success ratios of the SGA, CGAs, and PfGA [3] for the benchmark problems

GA	SGA	SGA*	CGA(WF)	CGA(NF)	PfGA [3]
J_1	6	6	6	6	20
scaling	linear	linear	exp.	exp.	none
$T_{\max}$	1,000	10,000	1,000	1,000	1,000**
F'_{th}	0.9	0.9	0.9	0.9	0.9
Shekel (K=2)	99%	99%	95%	100%	100%
(K=5)	5%	5%	5%	50%	37%
(K=10)	0%	0%	0%	0%	1.3%
F'_{th}	1.4	1.4	1.4	1.4	1.4
Langermann (K=5)	41%	47%	13%	35%	83%
(K=10)	0%	0%	0%	3%	1.7%

For each run out of 100, success was judged by whether F' exceeded F'_{th} before reaching the maximum number of generations, $T_{\max}$. The CGA(WF) is an original model with translation feedback by aa-tRNA, while the CGA(NF) is a modified model with no feedback. **The PfGA performed 300 runs and evaluated the same number of individuals, $N \cdot T_{\max} = 256{,}000$, as the SGA and CGAs for each run.

5 Evolvability Analysis

The reason why CGAs show good performance is thought to be the fact that during evolution, they adaptively change the binary-to-real value translation and optimize the genotype-to-phenotype mapping. This optimization makes the fitness landscape smoother and make it easy for evolution to find a path to an optimum solution. To clarify the mechanisms underlying the CGA and confirm the above characteristics, we introduced a measure for evaluating the smoothness of the fitness landscape, the basin size of the global optimum solution (B) in the binary genotype space, and we conducted experiments to measure B. The deceptive problems and Shekel's foxholes were analyzed in terms of this metric for evolvability.

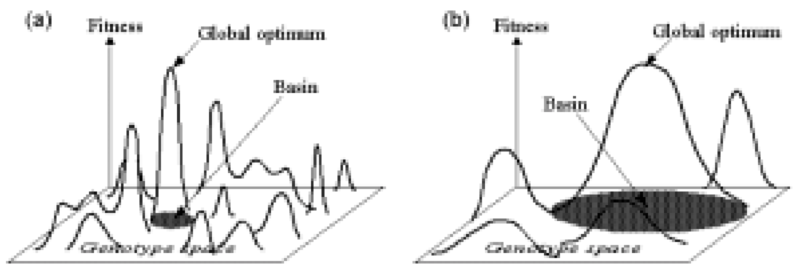

Fig. 3. Schematic of the fitness landscape and basin for a global optimum solution. (a) Evolvability is low, due to the small basin for the global optimum solution. (b) Evolvability is high due to the large basin.

If the fitness landscape is rugged (multipeaked) and the global optimum solution has a small basin in the genotype space (Fig. 3(a)), a mutation can hardly find this solution. If the fitness landscape is smooth and the optimal solution has a large basin (Fig. 3(b)), however, there is a strong possibility of evolution by a mutation finding a path to the global optimum solution and climbing the mountain. As shown by this argument, we can take the global optimum solution's basin size (B) in the binary genotypic space as a measure for a GA's evolvability. Although GAs utilize such genetic operations as crossover, which simultaneously changes multiple bits in a chromosome, we only focus here on the mutation's searching ability as a first-order approximation, and we evaluate B on a Hamming space in which two genotypes are regarded as adjacent if their Hamming distance is one.

5.1 Analysis with a Deceptive Problem

We first analyze the five-dimensional complex deceptive problem by using separation of variables. The calculation of basin size is based on the codon-to-amino translation table for each cell, which is built by the majority codon selection method, defined as follows. For a given codon, all aa-tRNAs within the current cell that contain the codon are listed. From this set, the most frequent amino value is selected as the translation value. If multiple amino acids share the highest frequency, one of them is randomly selected to determine the translation. Building the translation table in this manner provides a more accurate mapping between codon values and amino values, which increases the accuracy in calculating the basin size.

Figure 4 shows the average f value for each dimension until saturation, the average F value until saturation, the average basin size for each dimension over 250 generations, and the average basin size for all dimensions over 250 generations. From Fig. 4(d), we observe that the CGA is attempting to increase the total basin size by working on individual dimensions, but since the translation table is shared by all dimensions, optimizing a single dimension may have adverse effects on the other dimensions. For comparison, in Figs. 4(a-0) to (a-4), we also plot the average f values that are actually output from the cells. These results verify the significance of the basin size, whose increase for a given dimension greatly contributes to finding a global solution for that dimension.

Figure 5 shows the codon translation tables obtained after the saturation of CGA evolution, calculated by the majority codon selection method. Figure 5(b) shows the population average of the codon-to-amino value translation table, while Figs. 5(a-0) to (a-4) are the population averages of the codon-to-f value mappings calculated by using the codon-amino translation table for each cell. This figure indicates that although the fundamental codon-to-amino mapping (Fig. 5(b)) does not have a smooth landscape, the codon-to-f_k value mappings (Figs. 5(a-0) to (a-4)) do have relatively smooth landscapes, enabling most (about ten) codons to find a path to a global solution. With this landscape, the deceptive problem is transformed into an easier problem that does not deceive GAs.

5.2 Analysis with Shekel's Foxholes

Because the Shekel's foxholes problem is not variable-separable, we cannot analyze the fitness landscape for this function by using the same method as for the deceptive function. Instead, here we limit the Shekel's foxholes problem to two dimensions and directly analyze the $F(x)$ landscape as a function of codon pairs (we take $J_1 = 4$, so that we consider an eight-bit genotype space). In this setting, the basin size is defined as the number of codon pairs in each cell that can reach a global solution by traversing successively through the neighboring codons with the largest $F(x)$ values. The translation from a codon value to an amino value is determined by the same majority selection method as that used above for the deceptive problem, with a global solution determined by the criterion, $F(x) > 0.9$.

Figures 6(a) and (b) show the average $F(x)$ value and basin size $\bar{B}$, respectively, for the CGA solution to this problem. The basin size for generations 0 to 100 is similar to what would be achieved with a random genotype-to-phenotype translation table. From the 100th generation, however, the CGA suddenly begins to increase the basin size, and after the 130th generation, it is maintained at more than 60 codon pairs. This means that according to the fitness landscape optimized by the CGA, the basin for the global solution covers about one quarter of the whole genotype space.

6 Conclusion

We have developed a new biomolecular algorithm, a *chemical genetic algorithm* (CGA), in which several types of molecules react with each other in a cell. To demonstrate the effectiveness of the CGA, we applied the algorithm to a set of deceptive problems and benchmark problems. The results show that co-evolution between codes and code translations increases the basin size (i.e., it enhances the evolvability) and makes the population converge to the global optimum with a higher probability.

The present version of the CGA is only a simple actualization of molecular reactions in a cell. The molecular types are a minimal set for the translation of DNA information, and some other important molecules, such as enzymes (proteins catalyzing reactions or synthesizing amino acids), are omitted. We consider introducing these molecules to the CGA to be an important issue for the future. Specifically, in the present model, the amino values are never newly synthesized; they are initially created in the set of amino acids and are just reused. Accordingly, the number of different Aminos monotonously decreases during evolution of the CGA. Making amino values newly synthesized during evolution is a future problem to be tackled.

References

1. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: *Molecular Biology of the Cell, The Fourth Edition.* Garland Publishing, New York (1994)
2. Bersini, H., Doringo, M., Langerman, S., Seront, G., Gambardella, L.: Results of the first international contest on evolutionary optimization (1st ICEO). *Proc. of the IEEE Int. Conf. on Evolutionary Computation* (1996) 611-615
3. Sawai, H., Kizu, S.: Parameter-free Genetic Algorithms Inspired by 'Disparity Theory of Evolution.' In: *Parallel Problem Solving from Nature (PPSN-V)* (1998) 702-711
4. Suzuki, H., Sawai, H.: Chemical genetic algorithms - Coevolution between codes and code translation. In: Standish, R.K., Bedau, M.A., Abbass, H.A. (eds.): *Proc. of the Eighth Int. Conf. on Artificial Life VIII* (2002) 164-172
5. Sawai H., Suzuki H.: Chemical Genetic Algorithms - Coevolutionary Genotype-Phenotype Mapping by Modeling of Metabolism in Cell. In: *Proc. of the IEEE Int. Conf. on Evolutionary Computation* (2003) 639-646

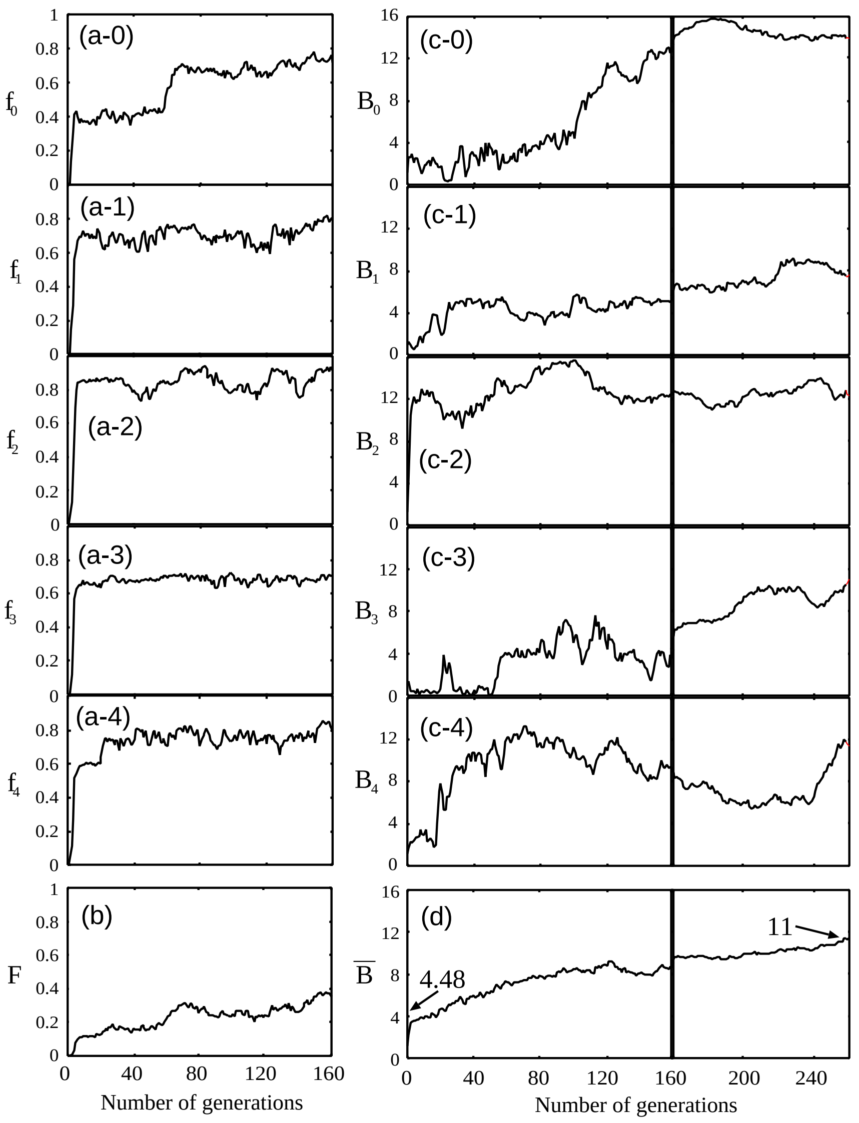

Fig. 4. Evolution of basin sizes and f-values for a typical run of the CGA. (a-0) to (a-4) represent the population-average f-value, f, for dimensions 0, 1, 2, 3, and 4, respectively, as functions of time from the start of evolution until saturation. (b) shows function F vs. time up to saturation. Functions F and f are as defined in Section 4.1. (c-0) to (c-4) illustrate the average basin size for dimensions 0, 1, 2, 3, and 4, respectively, until and after saturation. (d) shows the average of the five previous plots. The maximum basin size is 16.

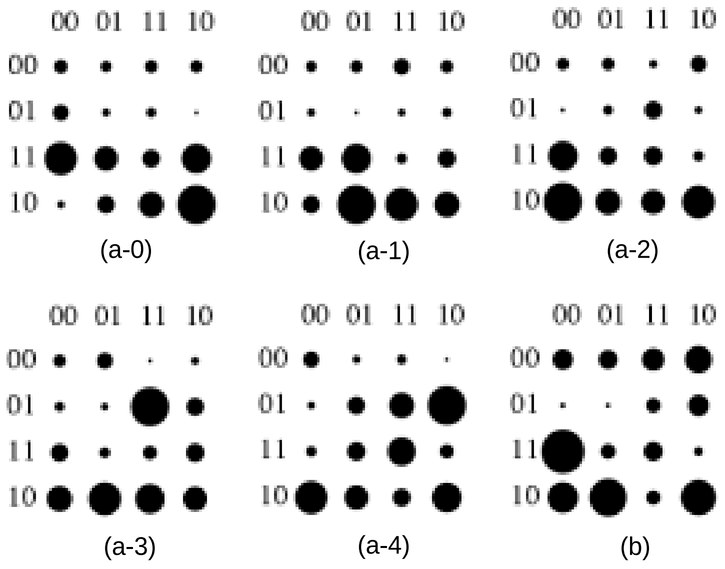

Fig. 5. Codon translation tables at the 200th generation of CGA evolution.

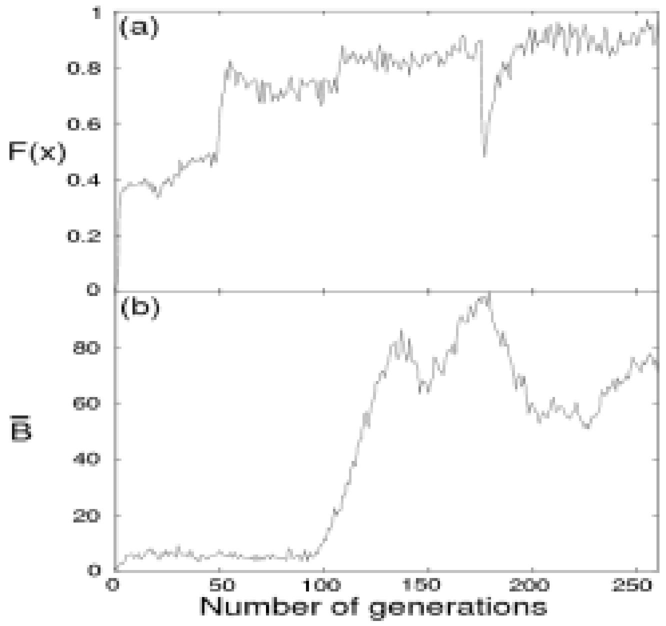

Fig. 6. Evolution of two-dimensional Shekel's foxholes for a typical run of the CGA. (a) Population-average F(x) value. (b) Population-average basin size, $\bar{\mathbf{B}}$. The maximum possible basin size is 256.