Chemical Genetic Programming – The Effect of Evolving Amino Acids

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

[1] ATR Human Information Science Labs., Kyoto, Japan
{wojtek,hsuzuki}@atr.jp,
[2] Communications Research Laboratory, Kobe, Japan
sawai@crl.go.jp

Abstract. A new method of genetic programming, named chemical genetic programming (CGP), which enables evolutionary optimization of the mapping from genotypic strings to phenotypic trees is proposed. A cell is evolved, and includes a DNA string that codes genetic information and smaller molecules for the mapping from DNA code to computational functionality. Genetic modification of a cell's DNA allows the DNA code and the genotype-to-phenotype translation to coevolve. Building an optimal translation table enhances evolution within a population while maintaining the necessary diversity to explore the entire search space.

1 Introduction

The genotype-to-phenotype mapping is a critical point in designing an evolutionary system. This mapping provides the building blocks that a system is allowed to work with in progressing towards its objective. To examine the meaning of this mapping, let us consider the biological system. In biological cells [1] information is derived from DNA to give each cell its functionality. This process is called translation. A series of metabolic reactions, catalyzed by several enzymes, translate genetic information into proteins, as shown in Fig. 1. A set of amino acids is prepared, mainly through assimilation of smaller inorganic compounds in plants, and through digestion of food in animals. Each amino acid is a biochemical building block, so together the amino acids form the fundamental set of functional units in a cell. Transfer RNA (tRNA) allows for DNA to specify which amino acids should be used in a cell. It is an adaptor between codons, or small DNA sequences representing instructions, and amino acids. An amino acid, the functional unit, and a tRNA, the informational unit, are combined into an aminoacyl-tRNA through a reaction catalyzed by an enzyme called aminoacyl-tRNA synthetase (ATS). ATS recognizes an identification sequence in tRNA, selects an appropriate amino acid that matches the identification sequence, and combines them. Proteins, which provide most of the machinery of a cell, are formed in the final stage of translation. The amino acid on an aminoacyl-tRNA joins a protein, thereby forming part of the protein's functionality.

An important feature of the biological translation from DNA, or genotype, into protein, or phenotype, is the use of *feedback (self-reference).* The genotype-to-phenotype translation is specified by the set of aminoacyl-tRNAs. The tRNA

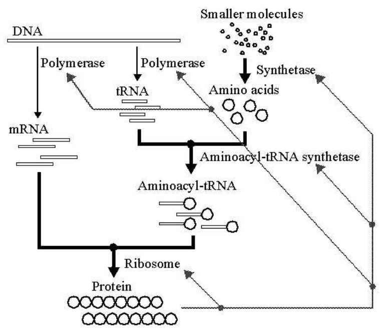
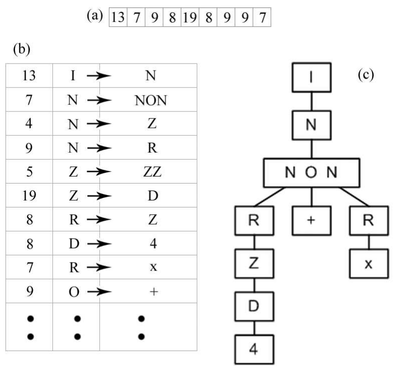


Fig. 1. Biochemical cells. Reactions for the translation of genetic information in a living cell. The rectangles represent informational molecules such as DNA and RNA, and the large circles represent functional units, amino acids.



13	I ➛	N
7	N ➛	NON
4	N ➛	Z
9	N ➛	R
5	Z ➛	ZZ
19	Z ➛	D
8	R ➛	Z
8	D ➛	4
7	R ➛	x
9	O ➛	+
⋮	⋮	⋮

Fig. 2. An example phenotypic tree created by the translation of a genotypic string, as shown in (a), using the translation table, as shown in (b), to generate the string 4+x, as shown in (c).

Table 1. The set of initial amino acids.

$I \rightarrow N$	$N \rightarrow NON$	$R \rightarrow x$	$Z \rightarrow D$	$O \rightarrow +$	$D \rightarrow 0$
	$N \rightarrow R$	$R \rightarrow Z$	$Z \rightarrow ZZ$	$O \rightarrow -$	$D \rightarrow 1$
	$N \rightarrow Z$	$R \rightarrow Z.Z$		$O \rightarrow *$	$D \rightarrow 2$
		$R \rightarrow N^N$		$O \rightarrow /$	$D \rightarrow 3$
					$D \rightarrow 4$
					$D \rightarrow 5$
					$D \rightarrow 6$
					$D \rightarrow 7$
					$D \rightarrow 8$
					$D \rightarrow 9$

in an aminoacyl-tRNA is copied directly from DNA, while the amino acid is created by metabolic reactions governed by proteins which are the result of a translation. Using this method, the translation set itself originates from DNA, and coevolution between the translation set and DNA is possible. Living cells may have evolved both genetic information and the basic translation relationships to enhance their own evolvability.

Borrowing the mechanisms described above, Suzuki et al. [2–5] recently proposed a new method of genetic algorithms named chemical genetic algorithms (CGA). CGA introduced a dynamic relationship between genotypic and phenotypic information and optimized the genotype to phenotype translation. This translation determines the performance of evolution, and by allowing it to be created through evolution rather than by human design, the system is allowed to build an optimal translation set.

In the studies of evolutionary computation on the other hand, Genetic Programming (GP) [6] has been successfully applied to various engineering problems. Starting with a population of randomly generated programs composed of predetermined programming ingredients, conventional GP evolves a population over a series of generations by breeding the fittest programs using genetic operations. The population has no knowledge of which ingredients are productive to evolution, and which hinder progress.

Extending the ideas of conventional GP, Grammatical Evolution (GE) [7, 8] was one of the first to distinguish between the genotype and phenotype in an individual. GE evolves a sequence of rule numbers that are translated, using a predetermined grammar set, into a phenotypic tree. Using the breeding techniques of conventional GP, GE has been shown to be an effective alternative to conventional GP.

Based on these previous studies, here we propose a new method of genetic programming called *chemical genetic programming* (CGP). Like GE, this method distinguishes between genotypic and phenotypic information, and translates from genotype to phenotype to evaluate a cell. In CGP however, the translation relation is dynamic, and the system is allowed to optimize both the combination of symbols in the genotypic string, as well as the translation set it uses to create a cell's phenotypic tree.

CGP aims to model biological cells by allowing the set of programming ingredients to change during evolution. By reproducing beneficial functions (rewriting rules) and eliminating destructive functions, CGP can create good programs with a higher probability. Artificial cells in chemical genetic programming are represented with a numerical string (DNA), transcription units (tRNAs), a rule set (amino acids), and a translation table (aminoacyl-tRNAs), enabling coevolution between the codes on DNA and their functionality. Cells are evolved using mutation, crossover, and molecular exchange operations, and evaluated against a given target function. The effectiveness of chemical genetic programming is demonstrated by observing distinguishing characteristics in solving a symbolic regression problem.

This paper presents the model and initial results of CGP. Following a detailed description of the CGP algorithm in Section 2, the experimental results are given in Section 3. Section 4 analyzes the results of this paper, and Section 5 gives closing remarks and describes further research involving CGP.

2 Model

CGP is based on translation from genotypic strings to phenotypic trees. Translation is explained, followed by a detailed explanation of the CGP algorithm, and finally a description of the parameters needed for a CGP run is provided.

2.1 Translation

Translation is the process of turning genotypic information into a phenotypic tree. The purpose is to obtain a representation of a cell that can be evaluated. Translation will be explained in the context of the demonstrative example shown in Fig. 2. The phenotypic tree of every cell is initialized in the same way, to the string *I*, a particular non-terminal symbol that serves as the root of every phenotypic tree. Fig. 2(c) shows this initialization. A non-terminal symbol is any symbol that can be rewritten to another string, and conversely, a terminal symbol is one that cannot be rewritten. The collection of grammatical rewriting rules, or translations, is stored in each cell's translation table. Each translation contains a non-terminal symbol that it translates from, and a string that it translates into, and a rule number by which it can be referenced. Fig. 2(b) shows the structure of a translation table.

Selecting the rule to apply is done by the cell's genotypic data. The genotype is a collection of rule numbers, as shown in Fig. 2(a). By applying the rules in the genotype from left to right, a cell is able to specify which translations to use, and in what order. In Fig. 2, rule number *13* at the leftmost of the genotype is selected first. All translations in the translation table that have *13* as their rule number are listed up, and a translation is randomly chosen among them. In this example, only one such rule appears, so the translation corresponding to rule *13* is applied. In this way, translations are chosen and applied using a frequency proportional selection method. If a selected rule cannot be applied since the current phenotypic string does not contain the specified non-terminal symbol, then another rule containing the same rule number is selected. If no translation containing a desired rule number can be applied, this rule number will simply be skipped.

The translation process is continued until all non-terminal symbols in the phenotype have been replaced, or until all rule numbers in the genotype have been applied. This termination condition makes it possible for non-terminal symbols to appear in the final phenotype. The final expression created in Fig. 2 is the string *4+x*.

2.2 Algorithm

CGP is a method for evolving a population of cells towards a target. After initialization, each generation cycle executes four operations: chemical reaction and evaluation, selection, DNA mutation, and DNA crossover and molecular exchange. Each operation is described in detail below.

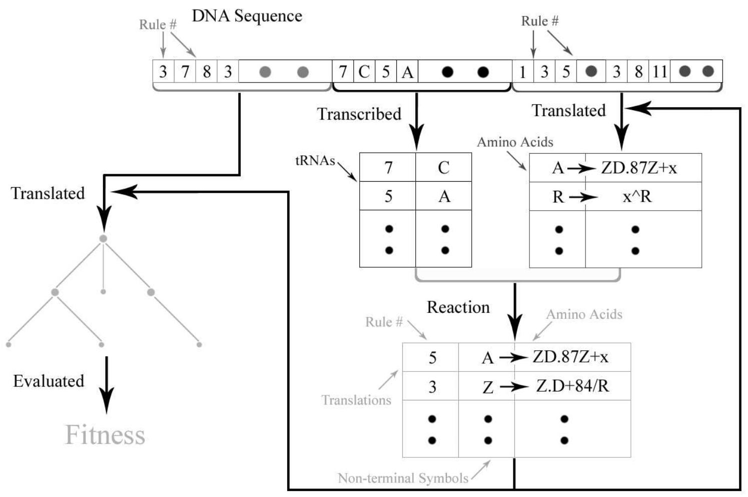


Fig. 3. Chemical reaction in a cell of CGP. A detailed description is provided in Sect. 2.2. The dots in each section of the figure indicate that the specified section, namely the DNA sequence, the collection of tRNAs, the collection of amino acids, and the collection of translations, contain more entries than the number shown in this figure. The size of each section is given in Table 2.

Initialization: A set of random individuals is created, each containing an empty translation table. Every individual starts with the initial amino acid library shown in Table 1. The DNA codes, or rule numbers, used to fill the DNA strings are selected at random between 0 and a parameterized maximum value.

The initial set of amino acids contains 6 non-terminal symbols, namely I, N, O, R, Z, and D, and 16 terminal symbols, namely x, ˆ, +, -, *, /, and the digits 0 to 9. All translations begin with the symbol I, and end when all non-terminal symbols have been replaced. Unlike conventional GP, here operators, such as * and ˆ, are treated as terminal symbols. The grammatical structure surrounding each operator is defined in other initial amino acids, such as $N \rightarrow NON$, rather than inside the operator itself. Each amino acid in the initial set is syntactically valid, which ensures that all amino acids created by translating elements from this set are also valid. Using this property, CGP is able to ignore the entire class of solutions that do not form executable structures.

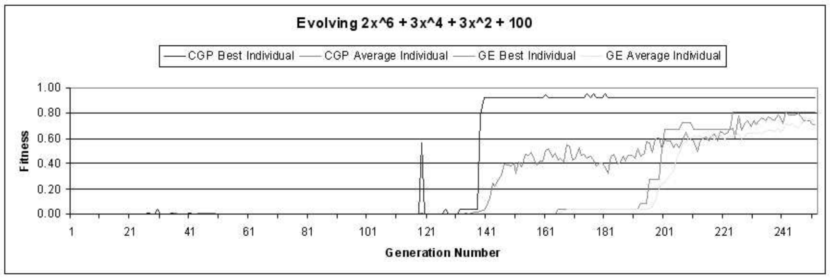


Fig. 4. The best and average normalized fitness values over the first 250 generations of evolution for both CGP and GE. CGP's best individual appears in generation 174. GE's best individual first appeared in generation 223.

Chemical Reactions and Evaluation: To determine the fitness of a cell, its genotypic data must be translated into phenotypic data. As described above, this process uses the cell's translation table. To determine the contents of the translation table, a cell undergoes chemical reactions. Initially, randomly selected entries are removed from a cell's amino acid pool, tRNA pool, and translation table. These removed elements will be replaced by newly created ones.

The tRNA pool, which is a collection of tRNA units, is repopulated using transcription. Each tRNA unit contains one rule number and one non-terminal symbol. Each cell's DNA contains a section of tRNA units, so transcription is simply copying tRNA from the DNA into the tRNA pool, as shown in the middle part of Fig 3.

Similarly, the amino acid pool is a collection of amino acids. Amino acids are grammatical rewriting rules, and can be viewed as a cell's building blocks. A new amino acid is created from the translation of a sequence of rule numbers on the right side of the DNA. The translation is the same as described above, with the exception the the selection of the root symbol. When creating amino acids, the root symbol is the non-terminal symbol in the translation specified by the first rule number. After translation, the new amino is created by copying the non-terminal symbol from the root of the tree to the left hand side and by copying the created expression (ex. $4{+}x$ in Fig. 2) to the right hand side of the rewriting rule. In this way, a newly created amino acid always contains a non-terminal symbol on the left-hand side and and a combination of terminal and non-terminal symbols on the right-hand side. Repopulating the amino acid table is done by performing the necessary number of translations and adding the resulting amino acids to the pool, as shown on the right-hand side of Fig. 3.

Once both pools have been repopulated, a cell is able to repopulate its translation table. An amino acid is chosen randomly from the amino acid pool, and a tRNA is selected randomly from the tRNA pool. The non-terminal symbol in each is compared, and if they are the same, a new translation is formed. This new translation takes the rule number from the tRNA, the common non-terminal symbol, and the rewriting string from the amino acid. This is repeated until the required number of translations have been created, with each being added to the translation table.

Finally, the leftmost portion of a cell's DNA is translated into a phenotypic tree, using this revised translation table, to allow the cell to be evaluated. This step is shown on the left-hand side of Fig. 3.

Using these translation reactions, the right-hand strings of amino acid rewriting rules become longer and more complex than previous ones. In order to adjust the growth rate, a mechanism has been added to allow amino acids from the initial amino acid table to re-enter a cell during the reaction process. A parameterized ratio, referred to as the initial amino acid ratio, is used to determine how many initial amino acids to reintroduce into the cell. Increasing the value of this ratio above zero allows the system to maintain a supply of the entire initial rule set. However, raising the value too much will cause the system to lose its ability to create and retain complex amino acids, so a middle value must be used. Introducing this mechanism stems from the biological analogy of a cell living inside another medium, and being able to accept amino acids from this medium.

Selection: Comparing an individual's phenotypic tree, or generated function, against a predetermined target function determines the individual's fitness. This value is used as the probability that the individual will be selected into the next generation, using roulette wheel selection. This selection method creates a wheel, with size equal to the sum of fitness values from the entire generation of individuals, where each individual represents an area on the wheel equal to its fitness value. The population of the next generation is the set of random selections from this wheel. When reproduced, the entire content of a mother cell is copied to the daughter cell.

DNA Mutation: Once individuals have been selected into a generation, their DNA units are genetically modified. Each individual is mutated immediately upon being selected to continue evolving. Each of its DNA codes are randomly replaced with another valid code with probability p_{m}.

DNA Crossover and Molecular Exchange: After mutation, individuals have probability p_{c} of being involved in a crossover operation. Single point crossover is performed between randomly selected pairs of DNA units. Half of each molecular unit, namely the translation table, tRNA pool, and amino acid pool, is exchanged between the two individuals during this operation.

Through evolution, beneficial amino acids will be propagated throughout the population, while destructive amino acids will be eliminated. The diversity of amino acids will gradually decrease, while the diversity of translations increases. The complexity of translations will grow in time, as productive combinations of amino acids can be merged into a single translation.

2.3 Parameters

The exact behavior of the model described above is determined by its set of parameters. This section will define the meaning of each parameter, with the value of each being shown in Table 2. The population size, probability of crossover, and probability of mutation are standard GP parameters that take on their conventional meaning. The values of these three parameters were selected to minimize the effective population size while maintaining diversity and guaranteeing the conservation of beneficial information in DNA. Using our translation method makes it possible to have an individual whose final phenotypic tree contains non-terminal symbols. In such a case the individual cannot be evaluated, so it is a part of the total population, but not part of the effective population. We experimentally determined that in order to have an effective population size of 143, we need a slightly larger actual population size.

The initial amino acid ratio is the proportion of amino acids to import from the initial amino acid table in a reaction. These amino acids are in addition to those created by translation. Genotypic length is the number of rule numbers to use for building a cell's phenotypic tree. The number of transcriptions determines how many "rule number-non terminal symbol" pairs to store in a cell's DNA. This also determines how many new tRNAs to add to the tRNA table at each generation. Translation length is the number of rule numbers to use to build a single amino acid. The number of translations indicates how many amino acids to create through translation in each reaction. This number also indicates how many amino acids to add to the amino acid pool at each generation. tRNA pool size and amino acid pool size determine the number of tRNAs and amino acids, respectively, to store in each pool at any one time. Translation table size is the maximum number of translations to store in the translation table. The maximum rule number sets the range of rule number values from 0 to this parameter value. New translations is the number of new translations to build at each generation by reacting the amino acid pool with the tRNA pool.

3 Experimental Results

To verify the effectiveness of CGP, we apply it to a symbolic regression problem. We chose a single variable mathematical expression to evolve. Specifically our target function is:

$$f(x) = 2x^6 + 3x^4 + 3x^2 + 100, \tag{1}$$

over the domain [-5, 5]. An individual's fitness value is defined as

$$fitness = exp(8 * exp(-d/50000000)), \tag{2}$$

where d is the sum of the differences between the generated function and target function squared at a finite set of points over the desired interval. We adopted this exponential function to enhance the fitness difference during the early stages of evolution where d has huge values.

The best result of 10 CGP runs is compared here against the best result of 10 GE runs, with both methods solving the same symbolic regression problem. Fig. 4 plots the normalized best and average fitness values for both CGP and GE, from the start of evolution until each method has found a good solution. A good solution is defined as having a normalized fitness value of 0.80 or more. For this run, CGP's best individual of all generations appeared in generation 174, approximately 60 minutes into evolution when running on a Pentium 4 3GHz desktop computer. GE's best individual appeared in generation 223, approximately 70 minutes into evolution running on the same computer. The GE parameters were the same as the CGP parameters, with the exception of the genotype length, which in GE was extended to 250. The rewriting grammar for GE was taken to be the same as the shown in Table 1.

The best generated functions are defined by Equation 3 and Equation 4 for CGP and GE, respectively. The normalized fitness values achieved by each method's best individual are 0.95 and 0.81 for CGP and GE, respectively, as calculated by Equation 2. The phenotypic tree for CGP's best solution and for GE's best solution are shown in Fig. 5 and Fig. 6, respectively.

$$f(x) = x^6 + x^6 + 501, \tag{3}$$

$$f(x) = x^6 * 1.9, \tag{4}$$

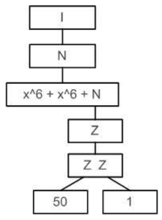


Fig. 5. The series of translations made for generating CGP's best individual in generation 174. A complex rewriting rule, $N \rightarrow x^6 + x^6 + N$, is used.

To examine the performances of CGP and GE more thoroughly, we compared 10 runs of CGP against 10 runs of GE. We selected two fitness milestones, and a timeline in which each should be reached. The fitness value 0.05 approximately corresponds to the function x^6 being discovered. The fitness value 0.80 indicates that the solution is very close to the target function. Table 3 shows how many out of the 10 runs achieved each milestone within the restricted time.

Finally we examine how much the growth of the amino acids rewriting rules affects evolution. We vary the value of the initial amino acid ratio, which was set to 0.3 in previous experiments, and observe the success probability of CGP as a function of this parameter. If we set this ratio to 1.0, we would eliminate the

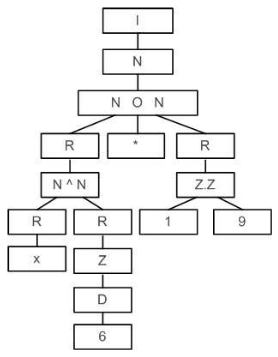


Fig. 6. The series of translations made for generating GE's best individual in generation 223.

Table 2. Parameter values used for the CGP experiments, as described in Sect. 2.3

Name	Value
Population size	170
Probability of crossover	0.7
Probability of mutation	0.007
Initial amino acid ratio	0.3
Genotypic length	50
Number of transcriptions	10
Translation length	6
Number of translations	9
tRNA pool size	200
Amino acid pool size	250
Translation table size	150
Maximum rule number	15
New translations	30

Table 3. Number of runs of CGP and GE that achieve milestone fitness values within a restricted number of generations.

Threshold / Generation	CGP	GE
0.05 / 300	8 / 10	4 / 10
0.80 / 750	7 / 10	2 / 10

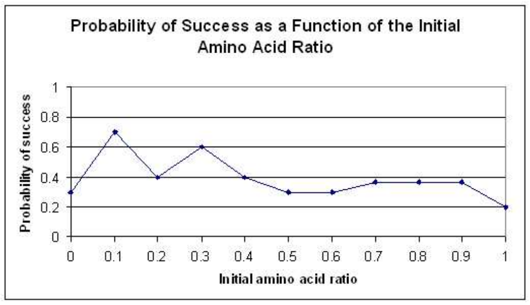


Fig. 7. Probability that a given initial amino ratio will produce a successful CGP run. This is based on a series of runs using the target function defined in Equation 1, and the success threshold 0.80 within 750 generations. Each result was calculated from 10 separate runs.

system's ability to evolve the translation table. In effect, we would be creating a system very similar to GE. Fig. 7 shows CGP's ability to evolve a good solution as a function of the initial amino acid ratio. We can see from this figure that the optimal ratio value is near 0.1, which sharply contrasts the algorithmically defined value of 1.0 that is used in GE.

4 Discussion

The results presented here indicate that CGP is a competent method for solving the tested symbolic regression problem. In this initial experiment we were able to observe the evolutionary characteristic that fundamentally differentiates our algorithm from others, namely the optimization of the translation table while achieving a good solution.

CGP's best phenotypic tree, as shown in Fig. 5, demonstrates the ability to evolve the fundamental genotype-to-phenotype relationship to improve a system's ability to find a good solution. By evolving complex rewriting rules, which solve the majority of a problem, CGP is able to thoroughly explore the search space in the immediate proximity of a particular solution. This indicates that the amino acid library is capable of being optimized to produce good individuals without converging to a single solution. CGP demonstrated its ability to diversify by continuing to explore solutions after a good result was found. Maintaining a variety of functional units in the amino acid pools, CGP can easily find an initial local optimum in early stages of evolution, and can escape from local optima later in evolution.

Another important point of CGP is the existence of *feedback*. CGP uses feedback to determine optimal content of the translation table. During its evolution, complex translations for creating subtrees may be combined into a single amino acid translation, allowing for more frequent use of productive rewriting rules.

This causes the number of translation required for a phenotypic tree to reach a complex solution to decrease over time, while the contents of the translation table become more complex.

5 Conclusion

We developed a new genetic programming method, called *chemical genetic programming*. By coevolving the DNA and fundamental genotype-to-phenotype translation table, CGP created an adaptive cell in which the final functional tree is created by the translation of symbol sequences in DNA using a set of evolved amino acid rewriting rules.

Further research will include applying CGP to more complex problems, more comparisons against other competent GP methods, and analyzing the mechanisms of CGP in terms of the enhancement of evolvability.

Acknowledgment

Dr. K. Shimohara of ATR actively supports this study. The first and second authors' work was funded in part by the NiCT of Japan. The second author's work was also supported by Doshisha University's Research Promotion Funds.

References

1. Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.D.: 1994. *Molecular Biology of the Cell, The Third Edition.* New York: Garland Publishing (1994)
2. Suzuki, H., Sawai, H.: Chemical genetic algorithms - Coevolution between codes and code translation. In: Standish, R.K., Bedau, M.A., Abbass, H.A. (eds.): *Proceedings of the Eighth International Conference on Artificial Life (Artificial Life VIII)* (2002) 164-172
3. Suzuki, H., Sawai, H.: Chemical genetic algorithms - Evolutionary optimization of code translation. In: Sugisaka, M., Tanaka, H. (eds.): *Proceedings of the Eighth International Symposium on Artificial Life and Robotics (AROB 8th '03)* (2003) Vol. 1 (2003) 172-175
4. Sawai, H., Suzuki, H.: Chemical genetic algorithms - Coevolutionary genotype-phenotype mapping by modeling of metabolism in cell. In: *Proceedings of the 2003 Congress on Evolutionary Computation (CEC-2003)* Vol. 1 (2003) 639-646
5. Suzuki, H., Sawai, H., Piaseczny, W.: Evolvability enhancement by the optimization of a chemical translation system - a case study. In: Dittrich, P., Kim, J.T. (eds.): *The 7th European Conference on Artificial Life (ECAL) Workshop Proceedings* (2003)
6. Koza, J.: 1992. *Genetic Programming*: On the Programming of Computers by Means of Natural Selection. London: MIT Press (1992)
7. Ryan, C., O'Neill, M., Collins, J.: Grammatical Evolution: Solving Trigonometric Identities. In: *Proceedings of the 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets (MENDEL)* (1998)
8. Ryan, C., O'Neill, M., Collins, J.: Grammatical Evolution: Evolving Programs for an Arbitrary Language. In: *Proceedings of the 1st European Workshop of Genetic Programming* (1998)