International Journal of Software Engineering & Research Methodology – IJSERM
Volume 1 : Issue 1
Publication Date : 09 January 2014

Application of Improved Grammatical Evolution to Stock Price Prediction

Hideyuki Sugiura, Takao Mizuno, Yukiko Wakita, Eisuke Kita

***Abstract*—Grammatical Evolution (GE), which is one of the evolutionary computations, aims to find function, program or program segment satisfying the design objective. This paper describes the improvement of the Grammatical Evolution according to Stochastic Schemata Exploiter (GE-SSE) and its application to symbolic regression problem. Firstly, GE-SSE is compared with original GE in symbolic regression problem. The results show that GE-SSE has faster convergence property than original GE. Secondly, GE-SSE is applied for the stock price prediction as the actual application of the GE-SSE.**

***Keywords*—Grammatical Evolution, Stochastic Schemata Exploiter, Symbolic Regression, Stock Price Prediction.**

I. Introduction

Grammatical Evolution (GE), which is one of evolutionary computations, is designed to find a function or a program or a program fragment that satisfies the given design objective [1-4]. Although the aim of GE is similar to the Genetic Programming (GP) [5], their algorithms are very different. In GE, the potential solutions (individuals) of the problem are defined as the bit-strings like Genetic Algorithm (GA). The genotype (bit-string) is translated into the phenotype (function or program) according to the translation rule. During GP search process, the genetic operators often generate the invalid phenotype. In GE, the use of the translation rule can avoid the invalid phenotype. Once the genotype is translated into phenotype, the fitness is estimated. The population of the potential solutions evolves toward better solutions by Simple Genetic Algorithm (SGA).

In this study, instead of the SGA, the Stochastic Schemata Exploiter (SSE) is employed for accelerating the convergence speed of the GE [6-8]. Stochastic Schemata Exploiter (SSE) comes from the Simple Genetic Algorithms (SGA) [9,10]. Their algorithms are very different. In SGA, the potential solutions (bit-strings) evolve to the optimal one by genetic operators such as selection, crossover, and mutation. The SSE search process also starts from the population of individuals. Sub-populations are defined from the whole population of individuals according to the descending order of their fitness. The set of common bits is extracted from the individuals in the sub-populations, which is named as common schemata. The set of uncommon bits in the bit-strings are replaced with randomly generated `0's and `1's for generating offspring. SSE uses the sub-population definition and the schemata extraction instead of selection and crossover employed in SGA.

Hideyuki Sugiura, Takao Mizuno, Yukiko Wakita, Eisuke Kita
Nagoya University, Graduate School of Information Science
Nagoya, Japan
E-mail: kita@is.nagoya-u.ac.jp

SSE has two interesting features. Firstly, the SSE convergence speed is faster than the SGA. Secondly, the crossover operation is not necessary in the SSE. Therefore, its performance does not depend on the crossover operation parameter. The aim of this study is to use SSE algorithm for improving the convergence property of the original GE. In the present algorithm, the update algorithm of individuals in the population is changed from SGA in the original GE to SSE. Symbolic regression problem is considered as the numerical example in order to discuss the convergence property.

The remaining part of this paper is organized as follows. The present algorithm is explained in section 2 and the results are shown in section 3. Finally, the conclusions are summarized again in section 4.

II. Grammatical Evolution with Stochastic Schemata Exploiter

A. *Algorithm*

The algorithm of Grammatical Evolution with Stochastic Schemata Exploiter (GE-SSE) is summarized as follows.

1. A translation rule is defined to translate genotype to phenotype.
2. An initial population is defined with randomly generated individuals.
3. Genotypes are translated to phenotypes according to the rule.
4. Fitness functions of phenotypes are estimated.
5. Convergence criterion is confirmed.
6. If the criterion is satisfied, the process is terminated. If not so, the process moves to the next step.
7. The population is updated by Stochastic Schemata Exploiter (SSE).
8. The process moves to step 3.

In original GE, the population update (Step 7 of the above algorithm) is done by Simple Genetic Algorithms (SGA). The present algorithm adopts Stochastic Schemata Exploiter (SSE) instead of SGA.

B. *Translation from Genotype to Phenotype*

The example of the translation rule is shown in Table 1. The translation rule (A) denotes that the symbol <expr> can be replaced with the symbols <expr><op><expr>, <num> or <var>. The symbol <expr> has three potential symbols to be

replaced <expr><op><expr>, <num> and <var>. The symbol <op>, <x> and <num> have four, two and only one potential symbols, respectively.

The translation of the symbol <expr> leads to the other symbols which can be replaced again. Such translation rule is named as the "recursive rule". The other symbols lead to the operators, variables and numbers which cannot be replaced any more. Such translation rule is named as the "terminal rule".

The translation process from genotype to phenotype can be summarized as follows.

1. A genotype is translated to a decimal number every n-bits.
2. A leftmost unused decimal number of the genotype is referred to as n_l.
3. The leftmost symbol in the character string is α, and the number of the candidate symbols for α is n_α.
4. The remainder n_r is calculated from n_l and n_α as $n_r = n_l \% n_\alpha$.
5. The symbol α is replaced with the n_r-th candidate symbol of the translation rule.
6. If the symbols exist, the process moves to step 2.

Table 1: BNF syntax in simple example

No	Rule
(A)	<expr> ::= <expr><op><expr> | <num> | <var>
(B)	<op> ::= + | - | * | /
(C)	<var> ::= x | y
(D)	<num> ::= 1

C. *Stochastic Schemata Exploiter*

The algorithm of Stochastic Schemata Exploiter (SSE) is summarized as follows [5-7].

1. Sub-populations are defined according to the descending order of the individual fitness.
2. Common schemata are extracted from the individuals in sub-populations.
3. The extracted schemata are composed of three characters; 0, 1 and *. New individuals are defined by randomly replacing * by 0 or 1.
4. The process moves to step 2 unless convergence criterion is satisfied.

(1) Sub-population Definition

The population P is composed of the individuals $c_1, c_2, \cdots, c_M$, which are numbered according to the descending order of their fitness. Therefore, the individual c_k denotes the k-th best individuals in the population P. The symbol S denotes the sub-population of the population P. When the individual c_k is excluded from S, a new population is represented as $S - c_k$. The operator ∪ denotes the union of sets.

When the number of the worst individual in the sub-population S is as $L(S)$, the individual $c_{L(S)}$ is the worst one in the sub-population S and thus, the individual $c_{L(S)+1}$ is worse by one rank than the individual $c_{L(S)}$. The following semi-order relation is held in the sub-population S of the population P.

1. When the individual $c_{L(S)+1}$ is added to a sub-population S, the new sub-population is defined as $S \cup c_{L(S)+1}$. The average fitness of the sub-population $S \cup c_{L(S)+1}$ is worse than that of the sub-population S.
2. When the individual $c_{L(S)}$ is replaced with the individual $c_{L(S)+1}$, the new sub-population is defined as $(S - c_{L(S)}) \cup c_{L(S)+1}$. The average fitness of the sub-population $(S - c_{L(S)}) \cup c_{L(S)+1}$ is worse than that of the sub-population S.

Sub-populations are defined according to their semi-order relation. The best sub-population is composed of the best individual c_1 alone, we have a first sub-population

$$S_1 = \{c_1\}. \quad (1)$$

When the individual c_2 is added to the sub-population $S_1 = \{c_1\}$ according to the semi-order relation, we have a second sub-population

$$S_2 = \{c_1, c_2\}. \quad (2)$$

When the individual c_1 in the sub-population S_1 is replaced with the individual c_2 according to the semi-order relation, we have a third sub-population

$$S_3 = \{c_2\} \quad (3)$$

When the individual c_3 is added to the sub-population $S_2 = \{c_1, c_2\}$ according to the semi-order relation, we have a fourth sub-population

$$S_4 = \{c_1, c_2, c_3\}. \quad (4)$$

When the individual c_2 in the sub-population S_2 is replaced with the individual c_3 according to the semi-order relation, we have

$$S_5 = \{c_1, c_3\}. \quad (5)$$

Similarly, sub-populations are defined according to the semi-order relation and the descending order of the individual fitness.

III. Numerical Examples

A. *Example 1*

Symbolic regression problem is considered as the numerical example. The exact function is given as follows.

$$f(x) = x^4 + x^3 + x^2 + x \quad (6)$$

The set of variable x_i is given as follows.

$$\{x_1, x_2, \cdots, x_{201}\} = \{-10.0, -9.9, \cdots, 10.0\} \quad (7)$$

The translation rule is shown in Table 2 and the start symbol is <expr>.

Table 2: Translation rule (Example 1)

No	Rule
(A)	<expr> ::= <expr><op><expr> | <var>
(B)	<op> ::= + | - | * | /
(C)	<var> ::= x | <num>
(D)	<num> ::= 1 | 2 |3 | 4 | 5 | 6 | 7 | 8 | 9

Table 3: Parameters (Example 1)

Max. generation	1000
Simulation times	100
Population size	300
Chromosome length	800
Radix conversion bit-length	8 bit

Table 4: Crossover and mutation rates (Example 1)

GE	Crossover rate = 0.6; Mutation rate = 0.075
SSE	Mutation rate = 0.3

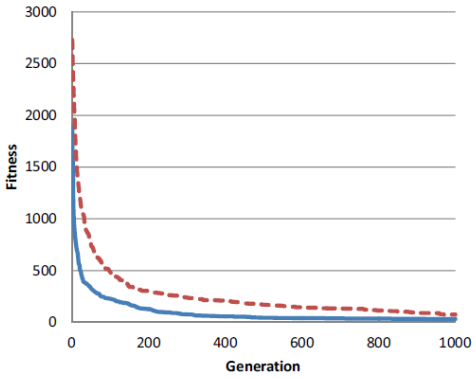

Figure 1: Convergence history of fitness (Example 1)

In the original GE, the individuals in the population are updated by SGA. The roulette selection is adopted and the selection probability P_x of the individual x is given by

$$P_x = 1 - \frac{f(x)}{\sum_{i=1}^{N} f(x_i)} \quad (8)$$

where N is total number of individuals. One-point crossover and elitist selection scheme are also used. Simulation parameters for GE and GE-SSE are shown in Table 3 and 4. Crossover and mutation rates for GE and GE-SSE were determined in advance by some numerical tests.

Figure 1 shows the convergence histories of the best individual fitness values in GE and GE-SSE. The figure is plotted with generation as horizontal axis and fitness value as vertical axis, respectively. This figure shows that GE-SSE could find the solution at 400 generation although the original GE found the similar one at 800 generation. Therefore, it is concluded that the convergence speed of GE-SSE is faster than that of GE.

Table 2 : Translation rule (Example 2)

No.	Rule
(A)	<expr> ::= <expr><op><expr> | <var>
(B)	<var> ::= <stock> | <num>
(C)	<op> ::= + | - | * | /
(D)	<stock> ::= y_{t-1} | y_{t-2} | y_{t-3} | y_{t-4} | y_{t-5}
(E)	<num> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

Table 3 : Parameters

Maximum generation	1000
Number of trials	100
Population size	300
Individual length	800
Number of elitists	3
Bit length for radix conversion	8 bit
Mutation rate	0.1

B. Example 2

GE-SSE is applied for symbolic regression problem of NIKKEI stock average. NIKKEI stock average values from October 1st 2010 to September 30th 2011 are taken as the training data for the problem. The function is applied for prediction of the NIKKEI stock average from October 3rd to November 30th, 2011.

The translation rule is listed in Table 2. The parameters y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4} and y_{t-5} denote the stock price one-, two-, three-, four- and five-days prefer of the day t. The symbol <expr> has two potential symbols; <expr><expr><op> and <var>. Similarly, the symbol <var>, <op>, <stock> and <num> have two, four, five and ten potential ones. The start symbol is <expr>. Parameters are shown in Table 3.

The fitness function is defined by the least square error of the real stock price and the predicted function values as follows

$$E = \frac{1}{N}\sum_{t=1}^{N}\sqrt{(y_t - \bar{y}_t)^2}. \quad (9)$$

The parameter N is the total number of days for stock price prediction. The function y_t and $\bar{y}_t$ denote the real stock price and the predicted price at the day t, respectively.

The simulations are performed 100 times. The best result among 100 simulations gives the following function.

$$\bar{y}_t = (y_{t-1} - 5) + \frac{(y_{t-3} - y_{t-5}) + 10}{y_{t-2}} + \frac{y_{t-5} - y_{t-4}}{5} \quad (10)$$

The stock price predicted by the above equation is compared with the actual stock price in Fig.2. The figure is plotted with the date as the horizontal axis and the stock price as the vertical axis, respectively.

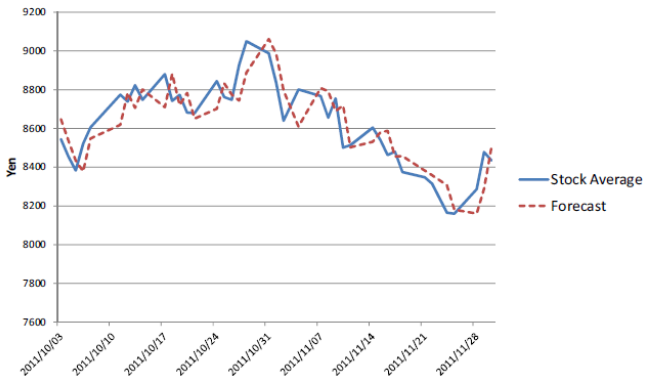

Figure 2: Comparison of stock price fluctuation (Example 2)

IV. Conclusion

This paper describes the improvement of the convergence property of Grammatical Evolution (GE). While the original GE uses Simple Genetic Algorithm (SGA) for updating the population, the present algorithm employs Stochastic Schemata Exploiter (SSE). Firstly, the algorithm was applied for symbolic regression problem. The results showed that the convergence speed of GE-SSE is faster than that of the original GE. The SSE tends to search the better solution than the best solution which has be found ever. Therefore, GE-SSE seems to show the faster convergence speed than the original GE. Secondly, the present algorithm was applied for stock price prediction problem. The result showed that the GE-SSE was applicable for this problem and there were some problems to be overcome.

References

[1] C. Ryan, J. J. Collins, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of 1st European Workshop on Genetic Programming*, 83-95. Springer-Verlag, 1998.

[2] C. Ryan and M. O'Neill. Crossover in grammatical evolution: A smooth operator? In *Proceedings of the European Conference on Genetic Programming*, 149-162. Springer-Verlag, 2000.

[3] M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349 -358, 2001.

[4] C. Ryan and M. O'Neill. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer-Verlag, 2003.

[5] J. R. Koza, *Genetic Programming II*, The MIT Press, 1994.

[6] N. A. Aizawa. Evolving SSE: A stochastic schemata explointer. In *Proc. 1st IEE Conference on Evolutionay Computation*, 525-529. IEEE, 1994.

[7] T. Maruyama and E. Kita. Estimation and extension of stochastic schemata exploiter. In *Data Mining, Text Mining and their Business Applications (Proceedings of Data Mining 2005)*, 45-54, 2005.

[8] T. Maruyama and E. Kita. Evaluation of extended stochastic schemata exploiter. In *Computer Aided Optimum Design in Engineering X (Proceedings of OPTI2007, USA)*, 45-54, 2007.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1 edition, 1975.

[10] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1 edition, 1989.