Computer Assisted Methods in Engineering and Science, **24**: 67–81, 2017.

Application of grammatical evolution to stock price prediction

Eisuke Kita[1], Hideyuki Sugiura[2], Yi Zuo[3], Takao Mizuno[2]

[1] *Graduate School of Informatics*
Nagoya University
Nagoya 464–8601, Japan
e-mail: kita@is.nagoya-u.ac.jp

[2] *Graduate School of Information Science*
Nagoya University
Nagoya 464–8601, Japan

[3] *Institute of Innovation for Future Society*
Nagoya University
Nagoya 464–8601, Japan

Grammatical evolution (GE) is one of evolutionary computation techniques. The aim of GE is to find the function or the executable program or program fragment that will find the optimal solution for the design objective such as the function for representing the set of given data, the robot control algorithm and so on. Candidate solutions are described in bitstring. The mapping process from the genotype (bitstring) to the phenotype (function or program or program fragment) is defined according to the list of production rules of terminal and non-terminal symbols. Candidate solutions are evolved according to the search algorithm based on genetic algorithm (GA). There are three main issues in GE: genotype definition, production rules, and search algorithm. Grammatical evolution with multiple chromosomes (GEMC) is one of the improved algorithms of GE. In GEMC, the convergence property of GE is improved by modifying the genotype definition. The aim of this study is to improve convergence property by changing the search algorithm based on GA with the search algorithm based on stochastic schemata exploiter (SSE) in GE and GEMC. SSE is designed to find the optimal solution of the function, which is the same as GA. The convergence speed of SSE is much higher than that of GA. Moreover, the selection and crossover operators are not necessary for SSE. When GA is replaced with SSE, the improved algorithms of GE and GEMC are named "grammatical evolution by using stochastic schemata exploiter (GE-SSE)" and "grammatical evolution with multiple chromosome by using stochastic schemata exploiter (GEMC-SSE)", respectively. In this study, GE-SSE is compared with GE in the symbolic regression problem of polynomial function. The results show that the convergence speed of GE-SSE is higher than that of original GE. Next, GE-SSE and GEMC-SSE are compared in stock price prediction problem. The results show that the convergence speed of GEMC-SSE is slightly higher than that of GE-SSE.

Keywords: grammatical evolution (GE), mapping process, stochastic schemata exploiter (SSE), symbolic regression problem, stock price prediction.

1. Introduction

Grammatical evolution (GE), which is one of the evolutionary computations [1–4], is designed to find a function or a program or a program fragment which satisfies the optimal solution of the design objective such as the function for representing the set of given data, the robot control algorithm and so on. The GE search process starts from defining the population of individuals. Individuals remember candidate solutions as the bitstrings or a set of binary numbers. The mapping process from genotype (bitstring) to phenotype (function or program) is done according to the production rules that connect the non-terminal symbols with the terminal symbols such as operators, variables and so on. Once an individual genotype is mapped to a phenotype, the fitness is calculated for

the phenotype of each individual. The search algorithm based on GA [5–7] is used for finding the optimal solution. While the aim of GE is the same as that of genetic programming (GP) [8, 9], the use of bitstring is similar to GA. There are three main key issues in GE: genotype definition, mapping process and search algorithm. For improving the genotype definition, Hara et al. presented "grammatical evolution with multiple chromosomes (GEMC)" [10]. On the list of the production rules, different production rules are defined for each non-terminal symbol. While the original GE uses single genotype (bitstring) for all production rules, GEMC uses different genotype (bitstring) for each production rule.

This study focuses on the improvement of GE and GEMC by using the search algorithm based on SSE [11–15], instead of the search algorithm based on GA. SSE, which is also one of the evolutionary computations, is designed to find the solution of the complicated objective function. Although the aim of SSE is similar to that of GA, their algorithms are very different. While GA finds the optimal solution by using genetic operators such as selection, crossover, and mutation, SSE uses schema operation and mutation. The search process of SSE also starts from the definition of individual population. Once every individual fitness in the population is estimated, the sub-populations are defined from the whole population according to their fitness and the semi-order relation of the sub-populations [11]. The sets of common binary numbers, which are called "schema", are extracted from the individuals in the sub-populations. Uncommon binary numbers are replaced with randomly generated '0's and '1's to generate offspring. The process is repeated until the convergence criterion is satisfied. GE and GEMC in which GA is replaced with SSE are named as "grammatical evolution by using stochastic schemata exploiter (GE-SSE)" and "grammatical evolution with multiple chromosome by using stochastic schemata exploiter (GEMC-SSE)", respectively. Best values for crossover and mutation rates of GE and GE-SSE are discussed in the symbolic regression problem. Next, the convergence performance of GE-SSE and GEMC-SEE is compared in the stock price prediction problem.

The remaining part of this paper is organized as follows. The algorithms of GE and GEMC are explained in Sec. 2. The application of SSE to GE is explained in Sec. 3. Numerical examples are shown in Sec. 4. The final conclusions are presented in Sec. 5.

2. Grammatical evolution

2.1. Original GE algorithm

The original GE algorithm is summarized above in Algorithm 1.

Algorithm 1. Original Grammatical Evolution

```
Set population size N.
Define mapping process with production rules, and terminal and non-terminal symbols.
Define fitness function which estimates the individual satisfying the design objective.
for all i such that 0 ≤ i < N do {Define initial population. }
  Define individual genotype with randomly generated bit-string.
end for
if The best individual does not satisfy convergence criterion. then
  for all i such that 0 ≤ i < N do {Calculate individual fitness. }
    Translate individual genotype to phenotype.
    Calculate fitness of individual phenotype.
  end for
  for all i such that 0 ≤ i < N do {Update population.}
    Update individual genotypes with the search algorithm based on GA.
  end for
end if
Output phenotype of best individual.
Terminate process.
```

Firstly, the mapping process is defined with the production rules and terminal and non-terminal symbols (Subsec. 2.2). Then, the fitness function is defined for estimating the individual satisfying the design objective. Initial population is defined with the individuals of randomly defined bitstrings.

If the best individual does not satisfy the convergence criterion, the individual genotype is translated into the phenotype and then the individual fitness is calculated (Subsec. 2.3). The individuals in the population are updated according to the search algorithm based on GA (Subsec. 2.4).

If the best individual satisfies the convergence criterion, the phenotype of the best individual is output and the process is terminated.

2.2. Symbols and production rules

A Backus-Naur form (BNF) grammar is consists of the tuple of N, T, P and S. N is the set of all non-terminal symbols, T is the set of all terminal symbols, P is the set of the production rules that map N to N or P, and S is the initial start symbol.

Using the tuple N, T and S as

```
N = { <expr> , <op> , <num> , <var> }
T = { + , - , * , / , x , y , 1 , 2 , 3 }
S = { <expr> }
```

the tuple P is shown in Table 1. The symbol "|" denotes the "or", which is the separations between the candidates. The production rule (A) denotes that <expr> can be replaced with <expr><op><expr>, <num> or <var>. In other words, <expr> has three candidates <expr><op><expr>, <num> and <var>. <op> has four candidates +, -, * and /. <x> has two candidates x and y, and <num> has three candidates 1, 2 and 3.

Table 1. Example of production rules. Four rules are defined for non-terminal symbols.

(A)	<expr> ::= <expr><op><expr>	(A0)
	| <num>	(A1)
	| <var>	(A2)
(B)	<op> ::= +	(B0)
	| -	(B1)
	| *	(B2)
	| /	(B3)
(C)	<var> ::= x	(C0)
	| y	(C1)
(D)	<num> ::= 1	(D0)
	::= 2	(D1)
	::= 3	(D2)

2.3. Mapping process from genotype to phenotype

2.3.1. Grammatical evolution

The mapping process from genotype to phenotype is summarized as follows:

1. The leftmost unused number is referred to as n_i.
2. The leftmost non-terminal symbol in the string is α, and the number of the candidates for α is n_α.

3. The remainder n_l is calculated by $n_l = n_i \% n_\alpha$ from n_i and n_α.

4. The non-terminal symbol α is replaced with the n_l-th candidate symbol of the symbol α.

5. If the non-terminal symbols exist, the process goes to step 1.

According to Table 1, the genotype

$$\{n_i\} = \{n_1, n_2, n_3, n_4, n_5, n_6\} = \{3, 1, 3, 0, 2, 1\} \tag{1}$$

is translated into the phenotype as follows:

1. The start symbol is α = S = `<expr>`.

2. The leftmost unused number is $n_1 = 3$. The symbol α = `<expr>` has three candidates, $n_\alpha = 3$. Since $n_l = n_1 \% n_\alpha = 3\%3 = 0$, the symbol α = `<expr>` is replaced with the 0th candidate `<expr><op><expr>` (A0).

3. The leftmost unused number is $n_2 = 1$. The leftmost non-terminal symbol α = `<expr>` has three candidates, $n_\alpha = 3$. Since $n_l = n_2 \% n_\alpha = 1\%3 = 1$, the symbol α = `<expr>` is replaced with the 1st candidate `<num>` (A1).

4. The leftmost unused number is $n_3 = 3$. The leftmost non-terminal symbol α = `<num>` has three candidates, $n_\alpha = 3$. Since $n_l = n_3 \% n_\alpha = 3\%3 = 0$, the symbol α = `1` is replaced with the 0th candidate `1` (D0).

5. The leftmost unused number is $n_4 = 0$. The leftmost non-terminal symbol α = `<op>` has four candidates, $n_\alpha = 4$. Since $n_l = n_4 \% n_\alpha = 0\%4 = 0$, the symbol α = `<op>` is replaced with the 0th candidate `+` (B0).

6. The leftmost unused number is $n_5 = 2$. The leftmost symbol α = `<expr>` has three candidates, $n_\alpha = 3$. Since $n_l = n_5 \% n_\alpha = 2\%3 = 2$, the symbol α = `<expr>` is replaced with the 2nd candidate `<var>` (A2).

7. The leftmost unused number of the genotype is $n_6 = 1$. The leftmost symbol α = `<var>` has two candidates, $n_\alpha = 2$. Since $n_l = n_6 \% n_\alpha = 1\%2 = 1$, the symbol α = `<var>` is replaced with the 2nd candidate `y` (C1).

8. Finally, the phenotype `1+y` is generated.

The process is summarized in Table 2.

Table 2. Example of mapping process in GE. The genotype $\{3, 1, 3, 0, 2, 1\}$ is finally mapped to $1 + y$ according to the production rules in Table 1.

Step	(n_i, n_α, n_l)	Selected symbol	Symbol evolution
1			`<expr>`
2	(3, 3, 0)	`<expr><op><expr>`	`<expr><op><expr>`
3	(1, 3, 1)	`<num>`	`<num><op><expr>`
4	(3, 3, 0)	`1`	`1<op><expr>`
5	(0, 4, 0)	`+`	`1+<expr>`
6	(2, 3, 2)	`<var>`	`1+<var>`
7	(1, 2, 1)	`y`	`1+y`

2.3.2. Grammatical evolution with multiple chromosomes

The production rules are shown in Table 1. The genotype is given as follows:

$$\{n_{ij}\} = \begin{Bmatrix} n_{A1} & n_{A2} & n_{A3} & n_{A4} \\ n_{B1} & n_{B2} & n_{B3} & n_{B4} \\ n_{C1} & n_{C2} & n_{C3} & n_{C4} \\ n_{D1} & n_{D2} & n_{D3} & n_{D4} \end{Bmatrix} = \begin{Bmatrix} 0 & 1 & 2 & 0 \\ 1 & 2 & 3 & 1 \\ 0 & 3 & 1 & 2 \\ 3 & 0 & 2 & 1 \end{Bmatrix} \begin{matrix} \leftarrow \text{Rule (A)}, \\ \leftarrow \text{Rule (B)}, \\ \leftarrow \text{Rule (C)}, \\ \leftarrow \text{Rule (D)}. \end{matrix} \tag{2}$$

The mapping process is summarized as follows:

1. The start symbol is α = `<expr>`.
2. The leftmost unused number for the symbol α = `<expr>` is $n_{A1} = 0$. The leftmost non-terminal symbol α = `<expr>` has three candidates, $n_\alpha = 3$. Since $n_l = n_{A1}\%n_\alpha = 0\%3 = 0$, α = `<expr>` is replaced with the 0th candidate `<expr><op><expr>`(A0).
3. The leftmost unused number for the symbol α = `<expr>` is $n_{A2} = 1$. The leftmost non-terminal symbol α = `<expr>` has three candidates, $n_\alpha = 3$. Since $n_l = n_{A2}\%n_\alpha = 1\%3 = 1$, α = `<expr>` is replaced with the 1st candidate `<num>` (A1).
4. The leftmost unused number for the symbol α = `<num>` is $n_{D1} = 3$. The leftmost non-terminal symbol α = `<num>` has three candidates, $n_\alpha = 3$. Since $n_l = n_{D1}\%n_\alpha = 3\%3 = 0$, the symbol α = `<num>` is replaced with the 0th candidate `1` (D0).
5. The leftmost unused number for the symbol α = `<op>` is $n_{B1} = 1$. The leftmost non-terminal symbol α = `<op>` has four candidates, $n_\alpha = 4$. Since $n_l = n_{B1}\%n_\alpha = 1\%4 = 1$, the symbol α = `<op>` is replaced with the 1st candidate `-` (B1).
6. The leftmost unused number for the symbol α = `<expr>` is $n_{A3} = 2$. The leftmost non-terminal symbol α = `<expr>` has three candidates, $n_\alpha = 3$. Since $n_l = n_{A3}\%n_\alpha = 2\%3 = 2$, the symbol α = `<expr>` is replaced with the 2nd candidate `<var>` (A2).
7. The leftmost unused number for the symbol α = `<var>` is $n_{C1} = 0$. The leftmost non-terminal symbol α = `<var>` has two candidates, $n_\alpha = 2$. Since $n_l = n_{C1}\%n_\alpha = 0\%2 = 0$, the symbol α = `<var>` is replaced with the 0th candidate `x` (A2).
8. Finally, the phenotype `1-x` is generated.

This process is summarized in Table 3.

Table 3. Example of mapping process in GEMC. The genotype (2) is finally mapped to $1 - x$ according to the production rules in Table 1.

Step	(n_i, n_α, n_l)	Selected symbol	Symbol evolution
1			`<expr>`
2	(0, 3, 0)	`<expr><op><expr>`	`<expr><op><expr>`
3	(1, 3, 1)	`<num>`	`<num><op><expr>`
4	(3, 3, 0)	`1`	`1<op><expr>`
5	(1, 4, 1)	`-`	`1-<expr>`
6	(2, 3, 2)	`<var>`	`1-<var>`
7	(0, 2, 0)	`x`	`1-x`

2.4. Search algorithm

The search algorithm is shown in Algorithm 2. In this study, the search algorithm uses one-point crossover, mutation, roulette selection and elite selection [5–7].

The population size and the number of elite individuals are referred to as N and N_e, respectively. N_e elite individuals are kept and then $N - N_e$ new individuals are generated by one-point crossover, mutation and roulette selection. Once parents are selected from the population according to the roulette selection, off-springs are generated from them according to one-point crossover and mutation.

Algorithm 2. Search Algorithm of GE

Set number of elite individuals N_e.
Keep N_e elite individuals.
for all i such that $0 \leq i < N - N_e$ **do**
  Select parent individuals from population by roulette selection.
  Generate offspring individuals by one-point crossover and mutation.
end for
Update population with N_e elite individuals and $N - N_e$ new individuals.

3. Application of stochastic schemata exploiter to grammatical evolution

3.1. GE-SSE algorithm

When the search algorithm based on GA in the original GE is replaced with the search algorithm based on SSE, the algorithm of GE-SSE is identical to that of the original SSE, except that GA is replaced with SSE as the search algorithm. Therefore, in Algorithm 1, the statement "Update individual genotypes with the search algorithm based on GA" is replaced with the statement "Update individual genotypes with the search algorithm based on SSE".

3.2. Stochastic schemata exploiter

The update algorithm of the population in SSE is summarized as follows:

1. Individuals are re-numbered in the descending order of their fitness.

2. When more than one individual has the same fitness value, they are numbered randomly.

3. Sub-populations are defined by some individuals selected from whole population, as shown in Subsec. 3.2.1.

4. Common schemata are extracted from the individuals in sub-populations, as shown in Subsec. 3.2.2.

5. Offspring are generated from the common schemata, as shown in Subsec. 3.2.3.

While SGA generates offspring from parents (individuals), SSE generates them from the common schemata of individuals. SSE tends to search for better solution from the set of the similar solutions to the best solution that has been found ever.

3.2.1. Semi-order relation and sub-population definition

The fitness of the individual c_i is estimated by the function $f(c_i)$. 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. The symbol S, which denotes the sub-population of the population P, is composed of the individuals $c_{s(1)}, c_{s(2)}, \cdots, c_{s(m)}$. The parameter $s(i)$ denotes the descending order of the fitness of individuals in the subpopulation S. The individual c' denotes worse one than all other in the subpopulation S:

1. When the individual c' is added to the subpopulation S, the subpopulation is S'. A first semi-order relation is given as follows:

$$f(S) \geq f(S'). \tag{3}$$

2. When the individual c' is replaced with $c_{s(m)}$, the subpopulation is S''. A second semi-order relation is given as follows:

$$f(S) \geq f(S''). \tag{4}$$

The use of semi-order relations gives the following order of sub-populations:

1. The first sub-population S_1 is composed of the best individual c_1 alone, $S_1 = \{c_1\}$.

2. The second sub-population S_2 is given by adding the individual c_2 to the sub-population S_1 according to the relation (3), $S_2 = \{c_1, c_2\}$.

3. The third sub-population is given by replacing the individual c_1 in the sub-population S_1 with the individual c_2 according to the relation (4), $S_3 = \{c_2\}$.

4. The fourth sub-population is given by adding the individual c_3 to the sub-population S_2 according to the relation (3), $S_4 = \{c_1, c_2, c_3\}$.

5. According to the similar procedure, the sub-populations are defined.

3.2.2. Common schemata

When the candidate solutions are defined in bitstrings, the common bits are kept and the uncommon bits are replaced with the character '`*`'s. In Fig. 1, the common bits in chromosome `1` and `2` are 1st, 2nd, 5th, 7th and 8th bits and the other bits are replaced with the character '`*`'s. Therefore, the common schema is '`10**0*11`'.

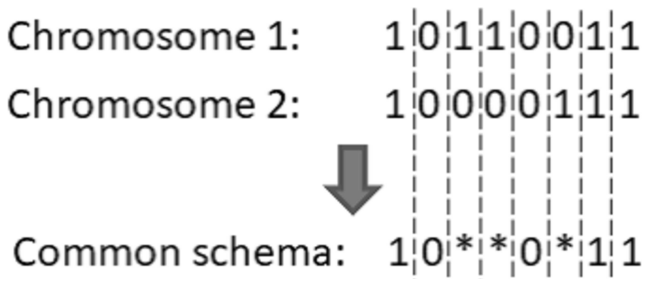

Fig. 1. Example of bitstrings and common schema. The common schema 10**0*11 is extracted from two individuals 10110011 and 10000111.

3.2.3. New individual generation

The extracted schemata are composed of three characters: '0', '1', and '∗'. So, the new individuals are generated by randomly replacing '∗' by '0' or '1'.

Offspring is generated from the schema shown in Fig. 1 as follows:

1. The schema '`10**0*11`' is shown in Fig. 1.
2. 3rd, 4th and 6th bits of the schema '`10**0*11`' are '∗'s.
3. '0', '1' and '1' are selected for 3rd, 4th and 6th bits of the schema.
4. Replacing the values of 3rd, 4th and 6th bits as '0', '1' and '1', respectively, the offspring is '`10010111`' (Fig. 2).

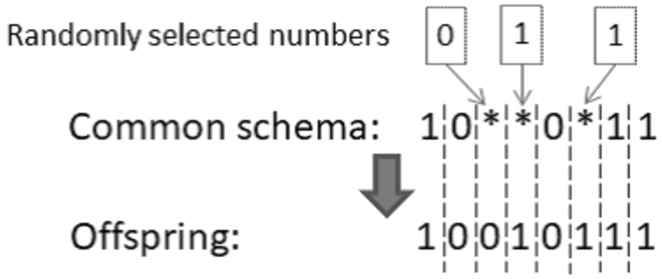

Fig. 2. Example of offspring generated from common schema. Offspring '10010111' is generated from the common schema '10**0*11'.

4. Numerical examples

4.1. Symbolic regression problem for polynomial function

4.1.1. Problem setting

The convergence properties of GE and GE-SSE are compared in the symbolic regression problem of the polynomial function.

The objective of this problem is to find the approximate function $\overline{f}$ that can express adequately the given data set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$. When the exact function f is given, the equation $y_i = f(x_i)$ is held.

The exact function is given as follows:

$$f(x) = x^4 + x^3 + x^2 + x. \tag{5}$$

The set of data for variable x_i is given as $\{x_i\} = \{-10.0, -9.9, \cdots, 9.9, 10.0\}$.

The set of all non-terminal symbols N, the set of all non-terminal symbols T and the initial start symbol S are given as follows:

```
N = { <expr>, <op>, <var>, <num> }
T = { +, -, *, /, x, 1, 2, 3, 4, 5, 6, 7, 8, 9}
S = { <expr> }
```

The production rules from genotype to phenotype are shown in Table 4.

Table 4. Production rules for symbolic regression problem of polynomial function.

(A)	<expr> ::= <expr><op><expr>	(A0)
	| <var>	(A1)
(B)	<op> ::= +	(B0)
	| -	(B1)
	| *	(B2)
	| /	(B3)
(C)	<var> ::= x	(C0)
	| <num>	(C1)
(D)	<num> ::= 1	(D0)
	| 2	(D1)
	| 3	(D2)
	| 4	(D3)
	| 5	(D4)
	| 6	(D5)
	| 7	(D6)
	| 8	(D7)
	| 9	(D8)

Fitness of each individual is given as follows:

$$E = \sqrt{\frac{1}{201}\sum_{i=1}^{201}(f_{(x_i)} - \overline{f}_{(x_i)})^2}. \tag{6}$$

The optimal solution is determined so as to minimize the fitness.

GA and SSE are applied for finding the function satisfying the data set. The parameters for GE are summarized in Table 5. Simulations are performed for different values of the crossover and mutation rates. Since the number of elites is 3, the first, second and third best individuals are kept in the population for the next generation. The mutation changes the value of the randomly selected gene from 0/1 to 1/0. Maximum generation is the number of iteration. Number of trials denotes the number of the independent runs. Population size is the number of individuals in the population. Chromosome length is the length of the individual genotype. The parameters for SSE are summarized in Table 6. One-point crossover and elite selection are not necessary for SSE.

Table 5. Parameters for GE.

Crossover rate	0.9, 0.8, 0.7, 0.6, 0.5
Mutation rate	0.5, 0.4, 0.3, 0.2, 0.1, 0.075, 0.05, 0.025, 0.01
Number of elites	3
Maximum generation	1000
Number of trials	100
Population size	300
Chromosome length	800

Table 6. Parameters for GE-SSE.

Mutation rate	0.5, 0.4, 0.3, 0.2, 0.1, 0.075, 0.05, 0.025, 0.01
Maximum generation	1000
Number of trials	100
Population size	300
Chromosome length	800

4.1.2. Effect of crossover and mutation rates

The effect of the crossover and mutation rates is discussed when GA is used for finding the solution. The average values of the fitness functions of the best individuals at the final generation are compared in Fig. 3 for different values of the crossover and mutation rates. The smallest fitness value is observed when the crossover rate is 0.6 and the mutation rate is 0.075. The effect of the mutation rate in GE-SSE is summarized in Fig. 4. The figure is plotted with the mutation rate as

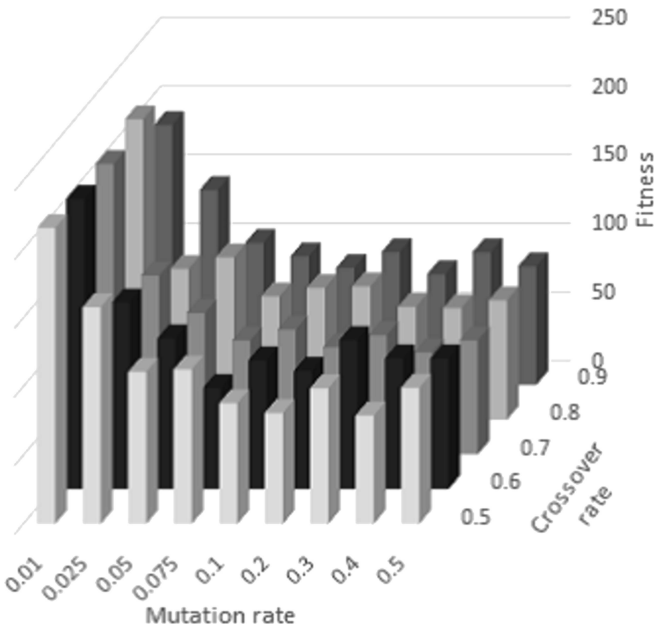

Fig. 3. Effect of crossover and mutation rates for GE.

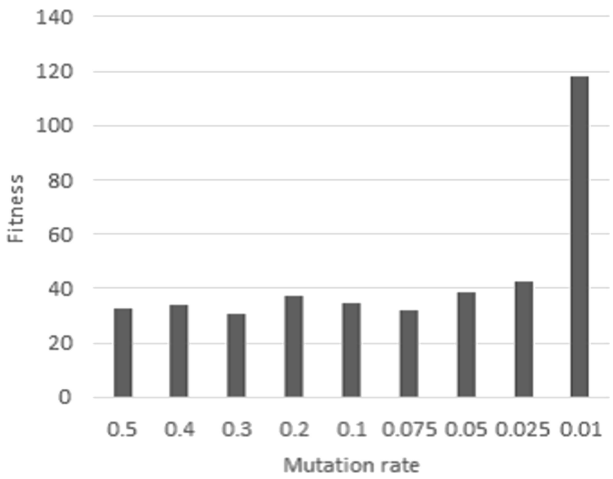

Fig. 4. Effect of mutation rate for GE-SSE.

the horizontal axis and the fitness as the vertical axis, respectively. The smallest fitness value is observed when the mutation rate is 0.3.

For GE, the crossover rate and the mutation rate are specified as 0.6 and 0.075, respectively. For GE-SSE, the mutation rate is specified as 0.3. The convergence histories of GE and GE-SSE are compared in Fig. 5. The figure shows that the convergence speed of GE-SSE is higher than that of GE and thus GE-SSE finds better solution than GE.

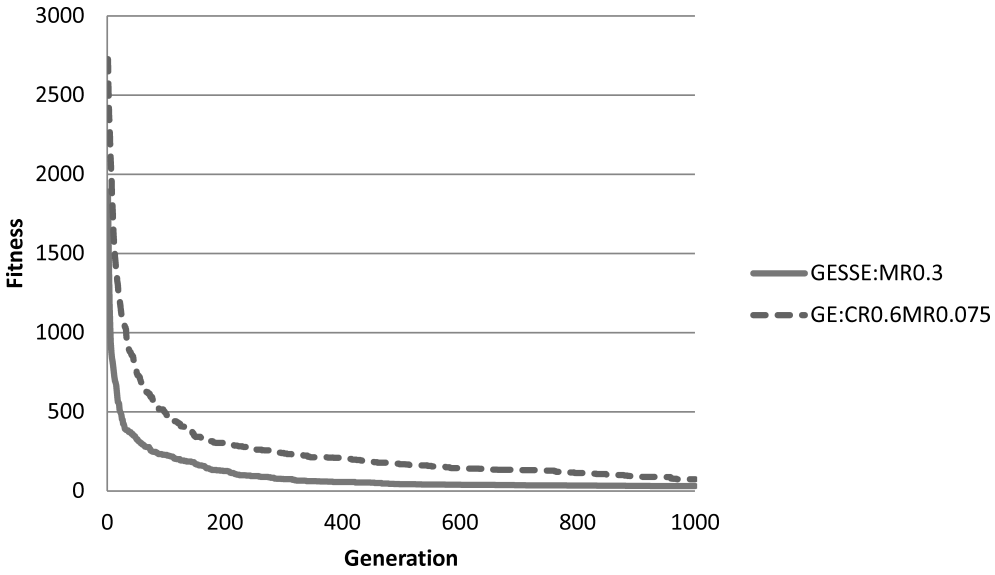

Fig. 5. Comparison of convergence histories of GE and GE-SSE.

4.2. Stock price prediction problem

4.2.1. Problem setting

The symbolic regression problem for stock price prediction is considered as the second numerical example. The prediction regression is generated from the Nikkei stock average data from October 1, 2010 to September 30, 2011. The generated regression is applied for the prediction of the stock fluctuation from October 3, 2011 to November 30, 2011.

The set of all non-terminal symbols N, the set of all non-terminal symbols T and the initial start symbol S are given as follows:

```
N = { <expr>, <var>, <op>,
      <stock>, <num>}
T = { +, -, *, /, y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4},
      1, 2, 3, 4, 5, 6, 7, 8, 9}
S = { <expr> }
```

The production rules from genotype to phenotype are shown in Table 7. The variable y_t denotes the stock price on the day t.

Fitness of each individual is given as follows:

$$E = \sqrt{\frac{1}{M}\sum_{t=1}^{N}(y_t - \overline{y_t})^2}. \tag{7}$$

Table 7. Production rules for stock price prediction problem.

(A)	<expr> ::= <expr><expr><op>	(A0)
	| <var>	(A1)
(B)	<var> ::= <stock>	(B0)
	| <num>	(B1)
(C)	<op> ::= +	(C0)
	| -	(C1)
	| *	(C2)
	| /	(C3)
(D)	<stock> ::= y_{t-1}	(D0)
	| y_{t-2}	(D1)
	| y_{t-3}	(D2)
	| y_{t-4}	(D3)
	| y_{t-5}	(D4)
(E)	<num> ::= 1	(E0)
	| 2	(E1)
	| 3	(E2)
	| 4	(E3)
	| 5	(E4)
	| 6	(E5)
	| 7	(E6)
	| 8	(E7)
	| 9	(E8)

The variables y_t and $\overline{y_t}$ denote the real stock price on the day t and the price estimated from the function predicted by GE, respectively. The parameter M is the total number of days in which Eq. (7) is estimated. The optimal solution is determined so that the fitness is minimized. Parameters are shown in Table 8.

Table 8. Parameters for GE for symbolic regression problem of stock price prediction.

Maximum generation	1000
Number of trials	100
Population size	300
Individual length	800
Number of elites	3
Mutation rate	0.5, 0.4, 0.3, 0.2, 0.1, 0.075, 0.05, 0.025, 0.01

4.2.2. Results and discussion

Firstly, GE-SSE is applied to the problem. Figure 6 shows the average values of the best individual fitness at the final generation. The fastest convergence speed is observed when the mutation rate is 0.1. The prediction function, which the best individual could find, is as follows:

$$\overline{y_t} = (y_{t-1} - 5) + \frac{(y_{t-3} - y_{t-5}) + 10}{y_{t-2}} + \frac{y_{t-5} - y_{t-4}}{5}. \tag{8}$$

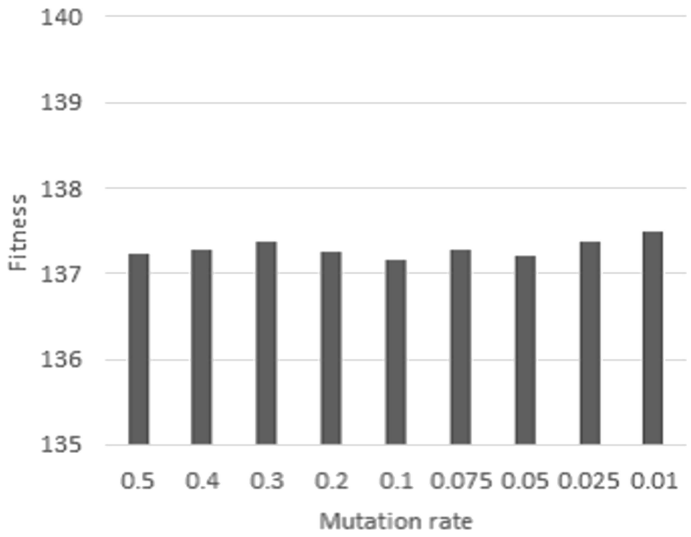

Fig. 6. Effect of mutation rate on average values of the best individuals at the final step in GE-SSE. The figure is plotted with the mutation rate as the horizontal axis and the average value of the best individuals' fitness, respectively.

Secondly, GEMC-SSE is applied to the problem. Figure 7 shows the average values of the best individuals fitness at the final generation. The figure is plotted with the mutation rate as the horizontal axis and the average value of the best individuals' fitness, respectively. The fastest convergence speed is observed when the mutation rate is 0.4. The prediction function, which the best individual could find, is as follows:

$$\overline{y_t} = (y_{t-1} + 4) + \frac{9y_{t-5}(y_{t-4} - y_{t-5})}{8y_{t-4} - 9y_{t-1}(y_{t-4} - y_{t-5})}. \tag{9}$$

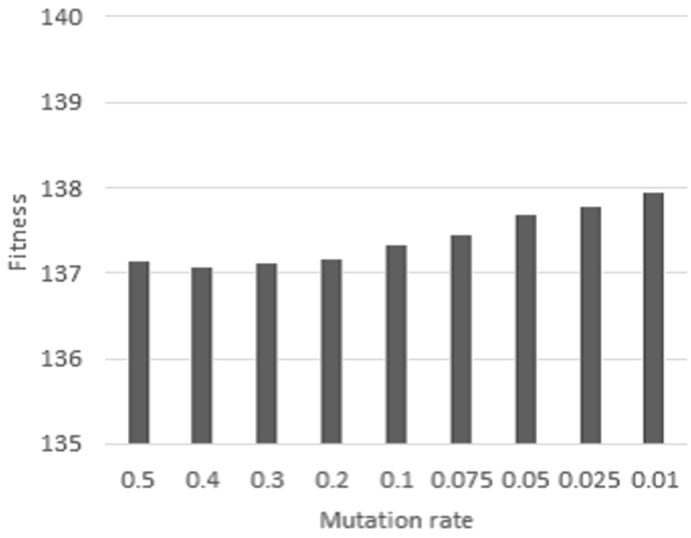

Fig. 7. Effect of mutation rate on average values of the best individuals at the final step in GEMC-SSE. The figure is plotted with the mutation rate as the horizontal axis and the average value of the best individuals' fitness, respectively.

The convergence histories of the best results of GE-SSE and GEMC-SSE are compared in Fig. 8. The generation and the fitness denote the iteration number and the fitness of the best individual at each generation, respectively. The figure shows that the convergence speed of GEMC-SSE is slightly higher than that of GE-SSE.

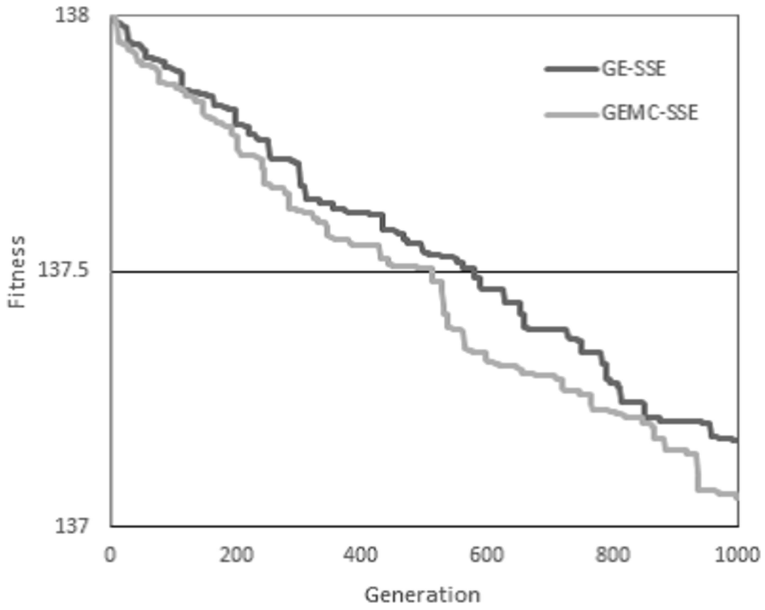

Fig. 8. Comparison of convergence histories of the best results of GE-SSE and GEMC-SSE. The figure is plotted with the generation as the horizontal axis and the fitness as the vertical axis, respectively.

Figure 9 compares the real stock price with the stock prices estimated from Eqs. (8) and (9). The figure shows that the stock prices estimated from Eqs. (8) and (9) are very similar, although the equations are different.

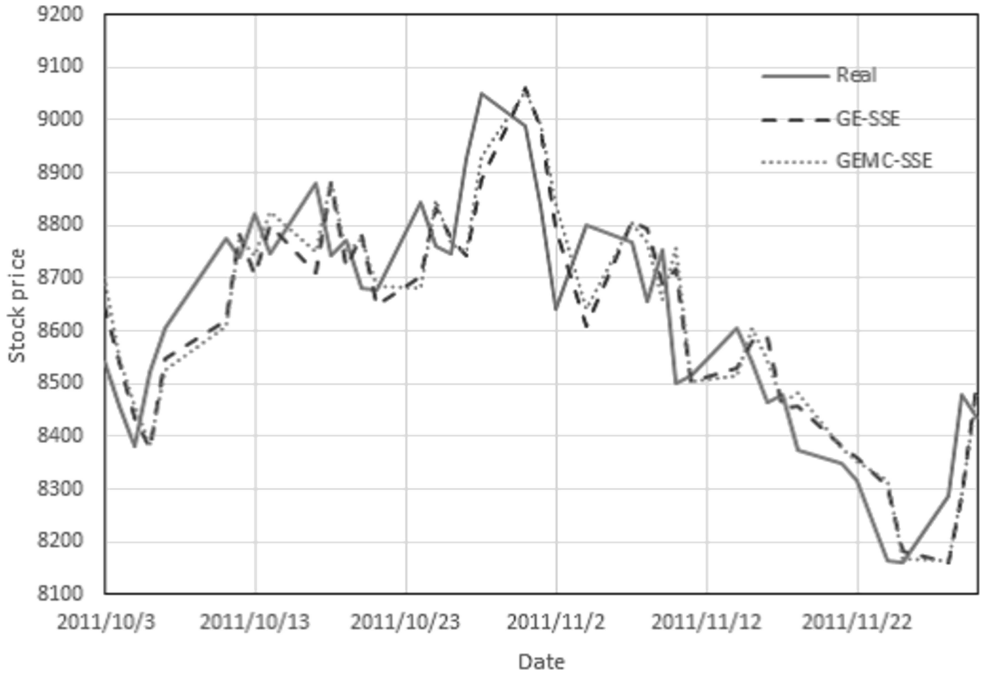

Fig. 9. Comparison of real and predicted stock prices. The figure is plotted with the date as the horizontal axis and the stock price as the vertical axis, respectively. The solid and the broken lines denote the real and predicted prices, respectively.

5. Conclusions

The aim of the GE is to find the function or the program which finds the optimal solution for the design objective such as the function for representing the set of given data, the robot control

algorithm and so on. Candidate solutions, which are defined as the bitstrings, are evolved with the simple genetic algorithm (SGA). In this study, SGA is replaced with SSE in order to improve the convergence speed of GE. The proposal algorithms are named as "grammatical evolution by using stochastic schemata exploiter (GE-SSE)" and "grammatical evolution with multiple chromosome by using stochastic schemata exploiter (GEMC-SSE)". Firstly, the best values for the crossover and mutation rates of GE and GEMC were discussed in the symbolic regression problem. The results show that the convergence speed of GE-SSE is higher than that of the original GE. Next, the convergence performance of GE-SSE and GEMC-SEE was compared in the stock price prediction problem. The convergence speed of GEMC-SEE is higher than that of GE-SSE, although their convergence properties are similar. Therefore, it is concluded that the use of SSE is effective for enhancing the convergence speed of GE.

References

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

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

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

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

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

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

[7] M.D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. The MIT press, 1st edition, 1999.

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

[9] J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane [Eds.]. *Genetic Programming III*. Morgan Kaufmann Pub., 1999.

[10] A. Hara, T. Yamaguchi, T. Ichimura, T. Takahama. Multi-chromosomal grammatical evolution. In: *Proceedings of 4th International Workshop on Computational Intelligence & Applications*, Okayama, Japan, 2008.

[11] N.A. Aizawa. Evolving SSE: A stochastic schemata exploiter. In: *Proc. of 1st IEEE Conf. Evol. Comp.*, pp. 525–529. IEEE, 1994.

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

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

[14] T. Maruyama, E. Kita. Cross-generational elitist selection SSE. In: *Proc. of the 7th World Congress of Structural and Multidisciplinary Optimization*, 2007. CD-ROM.

[15] T. Maruyama, E. Kita. Convergence analysis of cross-generational elitist selection SSE. In: *Proceedings of the 2008 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2008)*, 2008.