Computer Assisted Methods in Engineering and Science, **20**: 3–13, 2013.
Copyright © 2013 by Institute of Fundamental Technological Research, Polish Academy of Sciences
TWENTY YEARS OF THE CAMES

Application of grammatical evolution to coefficient identification problem in two-dimensional elastic problem

Takuya Kuroda, Hideyuki Sugiura, Eisuke Kita
Graduate School of Information Science
Nagoya University, Nagoya 464-8601, Japan
e-mail: kita@is.nagoya-u.ac.jp

Grammatical evolution (GE), which is a kind of evolutionary algorithms, is designed to find a function, an executable program or program fragment that will achieve a good fitness value for the given objective function to be minimized. In this study, GE is applied for the coefficient identification problem of the stiffness matrix in the two-dimensional elastic problem.

Finite element analysis of the plate with a circular hole is performed for determining the set of the stress and the strain components. Grammatical evolution determines the coefficient matrix of the relationship between the stress and strain components. The coefficient matrix is compared with Hooke's law in order to confirm the validity of the algorithm. After that, three algorithms are shown for improving the convergence speed of the original GE algorithm.

Keywords: grammatical evolution, Backus-Naur form (BNF), coefficient matrix, plane strain state.

1. Introduction

Genetic algorithms (GA) [1] and genetic programming (GP) [2] are very popular evolutionary computation techniques. Their objectives, however, are a little different. While the aim of traditional GA is to find the optimal or better solution of the optimization problem, GP is designed for finding the function representation, executable program or program fragment.

Grammatical evolution, shortly GE, was first presented by Ryan, Collins and O'Neill [3, 4]. The aim of GE is to find an executable program or program fragment that will achieve a good fitness value for the given objective function. The interesting feature of the GE is to represent function or program not by tree structure employed in GP but by binary numbers (bit-strings) in GA. The translation rules defined in the Backus-Naur form (BNF) are used for translating genotype (bit-string) to phenotype (function or program). A population of potential solutions evolves toward better solutions by using the genetic operators such as crossover, selection, mutation, and so on. Except for the use of translation rules, the GE algorithm is very similar to traditional GA. Therefore, the genetic operators designed for GA are available for GE.

GE is successively applied to the function identification problems [4, 5]. In the function identification problem, the data set is given and then, the function representing the data set is determined. In this study, GE is used for the identification of the coefficients of the stiffness matrix in the plane strain state when the set of stress and strain components is given. The stiffness matrix is defined in two- or three-dimensional matrix. The matrix components have to be determined so as to satisfy the relationship between the stress and strain components. Therefore, the stiffness matrix identification problem is more difficult than the ordinary function identification problem. The set of stress and strain components is calculated from finite element analysis of two-dimensional elastic body. Then, the coefficients of the stiffness matrix between the stress and strain components are determined by GE. The coefficient matrix is compared with Hooke's law. After that, three algorithms are shown for improving the convergence speed of GE. Finite element method is used for evaluating the stress and strain components [6–9]. Since it is a very popular tool in numerical analysis, it is very often employed for several identification and optimization problems [10–15].

The remaining part of this paper is organized as follows. In Sec. 2, the two-dimensional elastic problem and finite element analysis are explained briefly. The original GE algorithm and the improved algorithms are explained in Sec. 3 and 4, respectively. The numerical examples are shown in Sec. 5 and the results are summarized in Sec. 6.

2. Two-dimensional elastic problem

2.1. Basic relationship

The vectors of the displacement, traction, strain and stress components in the two-dimensional elastic problem are represented as $\boldsymbol{u}, \boldsymbol{t}, \boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}$, respectively;

$$\boldsymbol{u} = \{u_1, u_1\}^T, \tag{1}$$

$$\boldsymbol{t} = \{t_1, t_2\}^T, \tag{2}$$

$$\boldsymbol{\epsilon} = \{\epsilon_{11}, \epsilon_{22}, \epsilon_{12}\}^T, \tag{3}$$

$$\boldsymbol{\sigma} = \{\sigma_{11}, \sigma_{22}, \sigma_{12}\}^T, \tag{4}$$

where the subscripts 1 and 2 denote x_1- and x_2-axis, respectively. The subscript T denotes the transposition of the vector.

The relationship between the displacement and the strain components is given as follows:

$$\boldsymbol{\epsilon} = \boldsymbol{A}\boldsymbol{u}, \tag{5}$$

where $\boldsymbol{A}$ is defined as

$$\boldsymbol{A} = \begin{bmatrix} \partial/\partial x_1 & 0 \\ 0 & \partial/\partial x_2 \\ \partial/\partial x_2 & \partial/\partial x_1 \end{bmatrix}. \tag{6}$$

The relationship between the strain and the stress components are given as

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\epsilon}, \tag{7}$$

where $\boldsymbol{D}$ is defined as

$$\boldsymbol{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \dfrac{\nu}{(1-\nu)} & 0 \\ \dfrac{\nu}{(1-\nu)} & 1 & 0 \\ 0 & 0 & \dfrac{(1-2\nu)}{2(1-\nu)} \end{bmatrix}, \tag{8}$$

where E and ν denote Young's modulus and Poisson's ratio of the material, respectively. The relationship between the stress and the traction components are given as

$$t_i = \sigma_{ij} n_j, \tag{9}$$

where n_j denotes the x_j-component of the outer normal vector on the boundary.

2.2. Finite element formulation [7, 9]

The principle of the virtual work without the body forces is given as

$$\int_{\Omega} \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma} d\Omega = \int_{\Gamma_t} \delta\boldsymbol{u}^T \bar{\boldsymbol{t}} d\Gamma, \tag{10}$$

where Ω, Γ_u and Γ_t denote the domain occupied by the object under consideration, its displacement- and traction-specified boundaries and the whole boundary, respectively; that is $\Gamma = \Gamma_u \bigcap \Gamma_t$. The symbol δ and the superscript T denote the virtual changes and the transpose of the matrix or vector, respectively.

Discretizing the left-hand side of Eq. (10) into N_e finite elements, we have

$$\int_{\Omega} \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma} d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} \delta\boldsymbol{\epsilon}_e^T \boldsymbol{\sigma}_e d\Omega. \tag{11}$$

The displacement components at element e are approximated by the interpolation functions $\boldsymbol{N}$ with the nodal displacements $\boldsymbol{U}_e$;

$$\boldsymbol{u}_e = \boldsymbol{N}\boldsymbol{U}_e. \tag{12}$$

The stress and the strain components may then be expressed as

$$\boldsymbol{\epsilon}_e = \boldsymbol{A}\boldsymbol{u}_e = \boldsymbol{A}\boldsymbol{N}\boldsymbol{U}_e \equiv \boldsymbol{B}\boldsymbol{U}_e \tag{13}$$

and

$$\boldsymbol{\sigma}_e = \boldsymbol{D}\boldsymbol{\epsilon} = \boldsymbol{D}\boldsymbol{B}\boldsymbol{U}_e. \tag{14}$$

Substituting the above approximate expressions into Eq. (11) gives

$$\int_{\Omega} \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma} d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} \delta\boldsymbol{\epsilon}_e^T \boldsymbol{\sigma}_e d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} (\boldsymbol{B}\delta\boldsymbol{U}_e)^T \boldsymbol{D}\boldsymbol{B}\boldsymbol{U}_e d\Omega \equiv \sum_{e=1}^{N_e} \delta\boldsymbol{U}_e^T h_e \boldsymbol{K}'_e \boldsymbol{U}_e, \tag{15}$$

where h_e and $\boldsymbol{K}'_e$ denotes the thickness and the stiffness matrix at the element e, respectively.

Discretizing the right-hand side of Eq. (10) by N_l boundary elements gives

$$\int_{\Gamma_t} \delta\boldsymbol{u}^T \bar{\boldsymbol{t}} d\Gamma = \sum_{l=1}^{N_l} \int_{\Gamma_{t_l}} (\boldsymbol{N}\delta\boldsymbol{U})^T \bar{\boldsymbol{t}} d\Gamma \equiv \sum_{l=1}^{N_l} \delta\boldsymbol{U}_l^T \boldsymbol{f}'_l, \tag{16}$$

where $\boldsymbol{f}'_l$ denotes the equivalent nodal force vector at the boundary element l.

Substituting Eqs. (15) and (16) into Eq. (10) gives

$$\sum_{e=1}^{N_e} \delta\boldsymbol{U}_e^T h_e \boldsymbol{K}'_e \boldsymbol{U}_e = \sum_{l=1}^{N_l} \delta\boldsymbol{U}_l^T \boldsymbol{f}'_l \tag{17}$$

and

$$\delta\boldsymbol{U}^T \boldsymbol{K}\boldsymbol{U} = \delta\boldsymbol{U}^T \boldsymbol{f}, \qquad \boldsymbol{K}\boldsymbol{U} = \boldsymbol{f}, \tag{18}$$

where $\boldsymbol{K}$ and $\boldsymbol{f}$ denote the global stiffness matrix and the global equivalent nodal force vector, respectively.

3. Grammatical evolution

3.1. Original algorithm

The algorithm of an original grammatical evolution (GE) is simply summarized as follows:

1. The translation rules from genotype (bit-string) to phenotype (function or program) are defined in a Backus-Naur form (BNF) syntax.
2. An initial population of the individuals is defined with randomly generated bit-strings.
3. Bit-strings are translated into the function or the program according to the translation rules.
4. Fitness functions of individuals are estimated.
5. Genetic algorithms update the population of individuals.
6. The process is terminated if the criterion is satisfied.
7. Go to step 3.

The translation from genotype to phenotype is as follows:

1. A genotype (bit-string) is translated to a decimal number every n-bits.
2. The leftmost recursive symbol in the phenotype is referred to as α; The leftmost decimal in the genotype and the number of the potential symbols for α are n_l and n_α, respectively;
3. The remainder is calculated as $n_r = n_l \% n_\alpha$.
4. The symbol α is replaced with the n_r-th symbol of the candidate symbols.
5. If nonterminal symbols exist, go to step 2.

In the genetic programming (GP), the programs rapidly grow in size over time. This is called a "bloat" to overcome this difficulty, the maximum size of the programs is restricted in advance. The similar idea is applied to GE. The maximum size of the programs is restricted to $L_{\max}$.

3.2. From genotype to phenotype

We would like to explain the translation from genotype (bit-string) to phenotype (function) by a simple numerical example.

The example of BNF syntax is shown in Table 1. We notice that the symbol `<expr>` has three candidate symbols; `<expr><op><expr>`, `<num>` and `<var>`. This means that the symbol `<expr>` can be replaced with the symbol `<expr><op><expr>`, `<num>` or `<var>`. The symbol `<op>`, `<x>` and `<num>` have four, two and one candidate symbols, respectively.

We would like to explain how to translate the bit-string "1101001001" to the function `1+Y`. The bit-string is translated into the decimal number every 2-bits. The process is illustrated in Fig. 1.

1. A first 2-bit of the bit-string "1101001001" is "11", which is the decimal number $n_l = 3$. The start symbol $\alpha =$`<expr>` has three candidate symbols; $n_\alpha = 3$. The remainder of $n_l = 3$ with respect to $n_\alpha = 3$ is $n_r = n_l \% n_\alpha = 0$. Then, the symbol $\alpha =$`<expr>` is replaced with 0-th candidate symbol `<expr><op><expr>` (A0);

Table 1. BNF syntax in simple example.

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

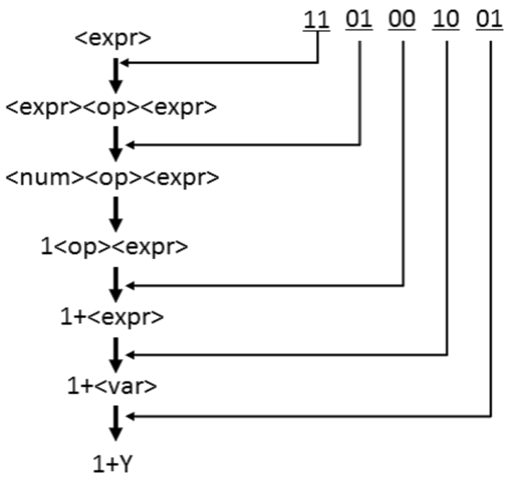

Fig. 1. Translation process from genotype to phenotype in simple example.

2. Next, we will consider the leftmost symbol $\alpha =$`<expr>` of the symbol `<expr><op><expr>`. A second 2-bit of the bit-string "1101001001" is "01", which is $n_l = 1$ in decimal number. The symbol $\alpha =$`<expr>` has three candidate symbols; $n_\alpha = 3$. The remainder of $n_l = 1$ with respect to $n_\alpha = 3$ is $n_r = n_l \% n_\alpha = 1$. The symbol $\alpha =$`<expr>` is replaced with `<num>` (A1) to generate the symbol `<num><op><expr>`;

3. The symbol `<num>` is replaced with the number `1` (D) to generate the symbol `1<op><expr>`;

4. A third 2-bit of the bit string "1101001001" is "00", which is $n_l = 0$ in decimal number. The symbol $\alpha =$`<op>` has four candidate symbols; $n_\alpha = 4$. The remainder of $n_l = 0$ with respect to $n_\alpha = 4$ is $n_r = n_l \% n_\alpha = 0$. The symbol $\alpha =$`<op>` is replaced with `+` (B0) to generate the symbol `1+<expr>`;

5. A fourth 2-bit of the bit string "1101001001" is "10", which is $n_l = 2$ in decimal number. The symbol $\alpha =$`<expr>` has three candidate symbols; $n_\alpha = 3$. The remainder of $n_l = 2$ with respect to $n_\alpha = 3$ is $n_r = n_l \% n_\alpha = 2$. The symbol $\alpha =$`<op>` is replaced with `<var>` (A2) to generate the symbol `1+<var>`;

6. The last 2-bit of the bit string "1101001001" is "01", which is $n_l = 1$ in decimal number. The symbol $\alpha =$`<var>` has two candidate symbols; $n_\alpha = 2$. The remainder of $n_l = 1$ with respect to $n_\alpha = 2$ is $n_r = n_l \% n_\alpha = 1$. The symbol $\alpha =$`<var>` is replaced with `Y` (C1) to generate the symbol `1+Y`.

4. Improved grammatical evolution

4.1. Difficulties of original GE

The original GE algorithm has two difficulties. One is related to the rule selection algorithm and another is to the selection probability of the candidate symbols.

4.1.1. Rule selection

The leftmost recursive symbol is referred to as α. The leftmost decimal number and the number of candidate symbols for α are referred to as n_l and n_α, respectively. Since a symbol is selected by the remainder $n_r = n_l \% n_\alpha$, the symbol selection is very sensitive to the variation of the decimal number n_l. Even when the value of n_l alters by only one, the symbol to be selected is changed. This may disturb the development of the better scheme included in the bit-strings. The scheme 1 is designed for overcoming this difficulty.

4.1.2. Selection probability of symbols

The original GE selects one symbol from the list of the candidate symbols. As shown in the example of Table 1, the symbol `<expr>` is replaced with the symbol (A0) `<expr><op><expr>`, (A1) `<num>` or (A2) `<var>`. The selection probabilities of the three symbols (A0), (A1) and (A2) are identical, 33% for each symbol. If, however, one candidate symbol is to be more promising than the others, the selection probability of the symbol should be enhanced.

The rules are classified into the recursive (non-terminal) and terminal rules. For example, in Table 1, the symbol (A) is recursive rule and the other symbols are terminal rules. The iterative use of recursive rule makes the phenotype (function or program) longer and more complicated. On the other hand, the terminal rule terminates the development of the phenotype. Since the functions of the recursive and the terminal rules are different, it is appropriate that the different selection probability is specified for the recursive and the terminal rules. The following scheme 2 and 3 are designed to control the selection probability of the recursive and the terminal rules, respectively.

4.2. Improvement of original GE

4.2.1. Scheme 1

In the original GE, the symbols are selected according to the remainder of the decimal number with respect to the total number of candidate symbols. The scheme 1 adopts the special roulette selection, instead of the remainder selection. The roulette selection is a popular selection algorithm in GA. In the scheme 1, the roulette selection probabilities for all candidates symbols are identical. The objective of the scheme 1 is to encourage the development of the better schemata.

We will consider a leftmost decimal as n_l, the leftmost nonterminal symbol as α, and the number of candidate symbols for α as n_α. The algorithm is as follows:

1. Calculate the parameter $s_\alpha = n_l/n_\alpha$.
2. Generate a uniform random number p $(0 < p \leq n_l)$.
3. If $(k-1)s_\alpha \leq p < ks_\alpha$, select k-th rule from the list of candidate symbols for α $(1 \leq k \leq n)$.

4.2.2. Scheme 2

In scheme 2, the selection probability of the candidate symbol in the recursive rule is controlled according to the depth of the tree structure. The maximum length of the programs is specified in advance. If the length of the programs is shorter than the maximum depth $L_{\max}$, the selection probability is increased. If not so, the probability is decreased.

The selection probability of the candidate symbol i in the recursive rule is calculated as

$$P_i^r = 1 - \frac{L}{L_{\max}}, \tag{19}$$

where L and $L_{\max}$ denote the depth and the maximum depth of the programs.

4.2.3. Scheme 3

In scheme 3, the selection probabilities of the candidate symbols in the terminal rules are controlled according to the total number of the candidate symbols included in all individuals in the population.

The total numbers of the candidate symbols in the terminal rules in all individuals are counted first. It is assumed that a terminal rule has N^N candidate symbols and that i-th candidate symbol occurs N_i times in all individuals. The selection probability P_i^N of the i-th candidate symbol is calculated as

$$P_i^N = \frac{N_i}{\sum\limits_{j=1}^{N^N} N_j}. \tag{20}$$

5. Numerical example

5.1. Problem

The use of GE determines the stiffness matrix $\boldsymbol{D}$ in Eq. (8) from the pairs of the stress vector $\boldsymbol{\sigma}$ and the strain vector $\boldsymbol{\varepsilon}$, which are determined by the finite element analysis.

A plate with a hole is shown in Fig. 2. When the plate is stretched in the horizontal direction, the stress analysis is performed by finite element method in order to determine the stress and strain distributions [9]. Triangle finite element discretization of the quarter part of the object domain is shown in Fig. 3. Parameters of finite element analysis are shown in Table 2.

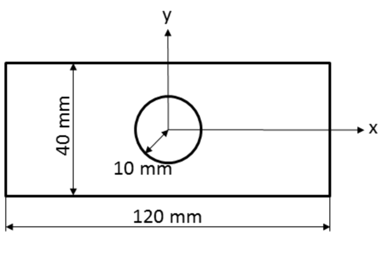

Fig. 2. Plate with a circular hole.

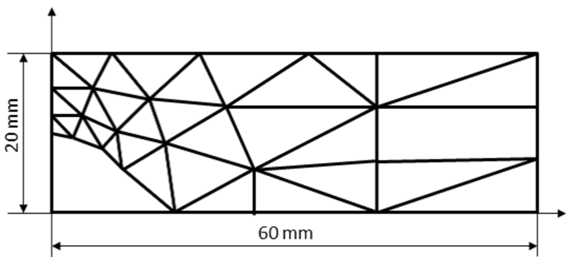

Fig. 3. Finite element discretization.

Table 2. Parameters of finite element analysis.

Element-type	Triangle constant strain element
Number of elements	34
Number of nodes	27
Young's modulus	$E = 10\,000$
Poisson ratio	$\nu = 0.3$

The stiffness matrix of the plane strain state is given as

$$\boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \dfrac{\nu}{(1-\nu)} & 0 \\ \dfrac{\nu}{(1-\nu)} & 1 & 0 \\ 0 & 0 & \dfrac{(1-2\nu)}{2(1-\nu)} \end{bmatrix}. \tag{21}$$

Since the stress and the strain components in the z-axis direction are independent of the other components, only the other coefficients in Eq. (21) are determined by GE. The stiffness matrix determined by GE is given as

$$\overline{\boldsymbol{D}} = \begin{bmatrix} \overline{D}_{11} & \overline{D}_{12} & 0 \\ \overline{D}_{21} & \overline{D}_{22} & 0 \\ 0 & 0 & \dfrac{E}{2(1+\nu)} \end{bmatrix}. \tag{22}$$

BNF syntax is listed in Table 3. The simulation parameters are shown in Table 4. The fitness is defined by the least square errors as

$$\mathit{fitness} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}[\{\overline{D}_{11}\varepsilon_x + \overline{D}_{12}\varepsilon_y - \sigma_x\}^2 + \{\overline{D}_{21}\varepsilon_x + \overline{D}_{22}\varepsilon_y - \sigma_y\}^2]}, \tag{23}$$

where M denotes the total number of elements. Fifty simulations are performed and the average values are estimated. In Eq. (23), the simulation process is summarized as follows:

1. Finite element analysis of a plate with a hole shown in Fig. 2 is performed to determine the stress component σ_x and σ_y and the strain component ε_x and ε_y.
2. An initial population of the individuals is defined with randomly generated bit-strings.
3. According to the translation rules shown in Table 3, bit-strings of individuals are translated to the coefficient $\overline{D}_{11}$, $\overline{D}_{12}$, $\overline{D}_{21}$ and $\overline{D}_{22}$.
4. Fitness functions of individuals are estimated by Eq. (23).
5. The evolution of population of individuals is maintained by genetic algorithm.
6. The process is terminated if the criterion is satisfied.
7. The process goes back to step 3.

Table 3. Translation rules.

(A)	`<expr> ::= <expr><op><expr>`	(A0)
	`| <var>`	(A1)
(B)	`<var> ::= <const>`	(B0)
	`| <num>`	(B1)
(C)	`<op> ::= +`	(C0)
	`| -`	(C1)
	`| *`	(C2)
	`| /`	(C3)
(D)	`<const> ::=` E	(D0)
	`|` ν	(D1)
(E)	`<num> ::= 1`	(E0)
	`| 2`	(E1)

Table 4. GE parameters.

Max. generation	500
Number of individuals	100
Chromosome length	100
Selection	Tournament
Tournament size	5
Number of elite individuals	1
Crossover	One-point
Crossover rate	0.9
Mutation rate	0.1
Radix conversion	Every 4 bit
Max. length of sentence	$MaxN = 100$

5.2. Result

The convergence histories of the fitness of the best individuals are shown in Fig. 4. The figure is plotted with the generation as the horizontal axis and the fitness as the vertical axis, respectively.

The convergence properties of the scheme 1 and 1+3 are similar to that of the original GE algorithm. The results by the use of scheme 1+2 and 1+2+3 show the faster convergence property than the ones obtained by the other schemes. Therefore, the scheme 2 is very effective for this problem and the combinational use of the scheme 2 and 3 is the most promising among them.

The scheme 1+2+3 determines the following stiffness matrix:

$$\overline{\boldsymbol{D}} = E \begin{bmatrix} 1+\nu & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & \frac{1}{2(1+\nu)} \end{bmatrix}. \tag{24}$$

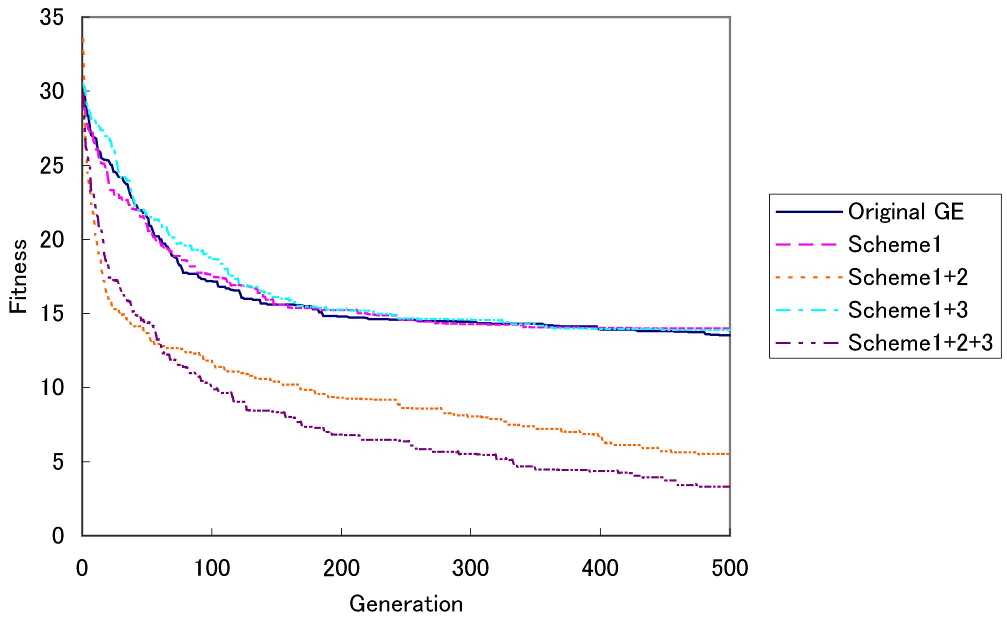

Fig. 4. Comparison of convergence speed.

When the parameters $E = 10000$ and $\nu = 0.3$ are substituted into Eqs. (8) and (24), the following matrices are obtained:

$$\boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} = \begin{bmatrix} 13461 & 5769 & 0 \\ 5769 & 13461 & 0 \\ 0 & 0 & 3846 \end{bmatrix}, \tag{25}$$

$$\overline{\boldsymbol{D}} = \begin{bmatrix} \overline{D}_{11} & \overline{D}_{12} & D_{13} \\ \overline{D}_{21} & \overline{D}_{22} & D_{23} \\ \overline{D}_{31} & \overline{D}_{32} & D_{33} \end{bmatrix} = \begin{bmatrix} 13000 & 5000 & 0 \\ 5000 & 10000 & 0 \\ 0 & 0 & 3864 \end{bmatrix}. \tag{26}$$

The relative error of the matrix element is estimated by the following equation:

$$e^D_{ij} = \frac{\|D_{ij} - \overline{D}_{ij}\|}{\|D_{ij}\|}. \tag{27}$$

The matrix element errors are shown in Table 5. Although Eq. (8) is different from Eq. (24) fairly, both matrix coefficients are relatively similar when comparing their values.

Table 5. Matrix element error.

$e^D_{ij}(\%)$	$j = 1$	$j = 2$	$j = 3$
$i = 1$	3.42	13.3	0
$i = 2$	13.3	25.7	0
$i = 3$	0	0	0

6. Conclusion

Grammatical evolution (GE), which is one of evolutionary computations, can represent tree structures such as functions and programs by the binary number. The use of the BNF syntax transforms binary numbers to functions or programs. After the original GE algorithm was shown, three improved algorithms were explained for improving the convergence property of the original GE. The improved algorithms were named as scheme 1, scheme 2 and scheme 3, respectively.

GEs were applied to determine the coefficients of the stiffness matrix in the plane strain state. The numerical results showed that GE could not find identical but similar stiffness matrix as Hooke's law. When the physical parameters were substituted into the stiffness matrix of Hook's law and the matrix predicted by GE, the numeric differences of their matrix components were relatively small. Therefore, the improvement of the GE prediction accuracy may depend on the fitness function definition and BNF syntax.

Finally, three schemes were compared in their convergence speed. Comparing the convergence properties of the schemes showed that the scheme 1+2+3 is the fastest and the scheme 1+2 is the second-fastest. Therefore, we can conclude that the scheme 3 is effective for this problem.

In this study, GEs were applied to determine the coefficients of the stiffness matrix in the plane strain state, which is one of special examples of function identification problems. We often encounter the similar problems in engineering application problems. Therefore, we would like to study some application of GEs to other engineering problems.

References

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

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

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

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

[5] A. Brabazon, M. O'Neill. *Biologically Inspired Algorithms for Financial Modelling.* Springer Verlag, 2006.

[6] K.-J. Bathe, E.D. Wilson. *Numerical Methods in Finite Element Analysis.* Prentice-Hall, 1976.

[7] K.-J. Bathe. *Finite Element Procedures in Engineering Analysis.* Prentice-Hall, 1982.

[8] D.S. Burnett. *Finite Element Analysis.* AT&T Bell Lab., 1987.

[9] O.C. Zienkiewicz, R.L. Taylor. *The Finite Element Method.* McGraw-Hill Ltd., 4th Edition, 1991.

[10] W. Grela, T. Burczynski. Evolutionary stress minimisation on a turbine blade shank. *Computer Assisted Methods in Engineering and Science*, **12**(2/3): 147–161, 2005.

[11] T. Burczynski, W. Beluch, A. Długosz, P. Orantek, A. Skrobol. Intelligent computing in inverse problems. *Computer Assisted Methods in Engineering and Science*, **13**(1): 161–206, 2006.

[12] A. Maniatty, N. Zabaras, K. Stelson. Finite element analysis of some inverse elasticity problems. *Journal of Engineering Mechanics*, **115**(6): 1303–1317, 1989.

[13] L. Houfek, P. Krejci, Z. Kolarova. Parameter identification of civil structure by genetic algorithm. In Ryszard Jablonski and Tomas Brezina [Eds.], *Mechatronics*, pp. 515–521, Springer Berlin Heidelberg, 2012.

[14] S. Dhandole, S.V. Modak. A constrained optimization based method for acoustic finite element model updating of cavities using pressure response. *Applied Mathematical Modelling*, **36**(1): 399–413, 2012.

[15] D. Moens, M. Hanss. Non-probabilistic finite element analysis for parametric uncertainty treatment in applied mechanics: Recent advances. *Finite Elements in Analysis and Design – Special Issue on Uncertainty and Structural Dynamics*, **47**(1): 4–16, 2011.