LETTER

A Conditional Dependency Based Probabilistic Model Building Grammatical Evolution

Hyun-Tae KIM[†], **Hyun-Kyu KANG**[††], ***Nonmembers*, *and* Chang Wook AHN**[†a)], ***Member***

SUMMARY In this paper, a new approach to grammatical evolution is presented. The aim is to generate complete programs using probabilistic modeling and sampling of (probability) distribution of given grammars. To be exact, probabilistic context free grammars are employed and a modified mapping process is developed to create new individuals from the distribution of grammars. To consider problem structures in the individual generation, conditional dependencies between production rules are incorporated into the mapping process. Experiments confirm that the proposed algorithm is more effective than existing methods.

key words: *grammatical evolution, probabilistic modeling, context-free grammars, automatic program generation*

1. Introduction

Evolutionary algorithms (EAs) employ the Darwinian principle of evolution for solving optimization problems. In order to discover the optimal solutions(s), they perform a population-based search by repetitively applying genetic operators (i.e., selection and recombination) to candidate solutions (i.e., population). So far, EAs have achieved much success in diverse areas, such as automatic programming. Of particular interest, grammatical evolution (GE) is an EA-based alternative to automatic programming methodologies. GE employs a population of linear genotypic integer strings, which are transformed into functional phenotypic programs by a genotype-to-phenotype mapping process. This process is influenced by context-free grammars (CFGs) that can contain domain knowledge to bias the form of possible phenotypic solutions. The grammars also implicitly provide a sort of instruction to generate program codes that are syntactically valid [1]–[3].

In EAs, the recombination of candidate solutions is very important since the operation is highly related to the preservation of partial information (called building blocks) of the optimal solution(s). In this respect, GE also applies crossover and mutation to the genotypic strings in the evolution process. However, existing methods often lose information when propagating the building blocks due to their random exchange of the contents of genotype. For this reason, estimation of distribution algorithm (EDA) has been proposed as a promising solution to realize the conservation of building blocks. EDA is a stochastic optimization technique that implements search mechanisms by building and sampling a probabilistic model of promising candidate solutions. It denotes that EDA can preserve the building blocks efficiently due to its ability to handle the distribution of population, rather than a tricky manipulation [4]. In this sense, probabilistic model building grammar evolution (PMBGE) has received a great attention of late, which is a variant of GE aimed to generate program codes by means of the distribution of given grammars [3]. In contrast with the conventional GE, PMBGE exploits EDA mechanisms as a search engine.

This paper presents a new approach to GE in conjunction with probabilistic models. To this end, probabilistic context-free grammars (PCFGs) are employed and the conditional dependencies between production rules are considered. Along with the distribution of production rules of the given grammars, the proposed approach can effectively discover an appropriate probabilistic model on the target problem.

2. Proposed Approach

In general, the context-free grammar (called BNF grammar) consists of a set of symbols: *non-terminals* and *terminals*. The non-terminals that do not appear in the program codes are replaced with other symbols. On the other hand, the terminals are visible from the outcomes of the mapping process; thus, variables or operators are represented as terminal symbols. Figure 1 depicts an example of context-free grammar and mapping process in the conventional GE. In this example, the grammar has three production rules on how a non-terminal is replaced with other symbols (called RHS).

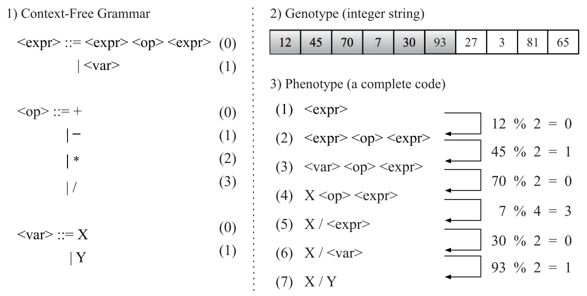

Fig. 1 An example of CFG and mapping process in GE

Manuscript received January 7, 2016.
Manuscript revised March 30, 2016.
Manuscript publicized April 11, 2016.

[†]The authors are with Department of Computer Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea.
[††]The author is with Department of Computer Engineering, Konkuk University, Chungju 27478, Republic of Korea.
a) E-mail: cwan@skku.edu (Corresponding author)

DOI: 10.1587/transinf.2016EDL8004

Algorithm 1 Conditional Dependency based PMBGE

```
Require: population size $N$, grammar $G$, probabilistic model $\mathcal{M}_G$
1: Randomly generate a population of size $N$, $P \leftarrow mapping(\mathcal{M}_G)$
2: while ‘Termination condition is not satisfied’ do
3:     Evaluate fitness values of population $P$
4:     Choose $N/2$ promising individuals ($P_S$) by tournament selection
5:     Estimate a distribution of $P_S$ considering conditional dependencies
6:     Update the probabilistic model ($\mathcal{M}_G$) on production rules in $G$
7:     Create $N/2$ offspring, $P_O \leftarrow mapping(\mathcal{M}_G)$
8:     Replace the population, $P \leftarrow P_S \cup P_O$
9: end while
```

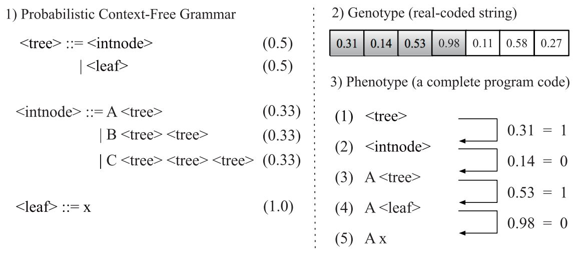
1) Probabilistic Context-Free Grammar
<tree> ::= <intnode> (0.5)
| <leaf> (0.5)
<intnode> ::= A <tree> (0.33)
| B <tree> <tree> (0.33)
| C <tree> <tree> <tree> (0.33)
<leaf> ::= x (1.0)
2) Genotype (real-coded string)
0.31 0.14 0.53 0.98 0.11 0.58 0.27
3) Phenotype (a complete program code)
(1) <tree>
(2) <intnode>
(3) A <tree>
(4) A <leaf>
(5) A x
0.31 = 1
0.14 = 0
0.53 = 1
0.98 = 0

Fig. 2 An example of PCFG and mapping process

Moreover, GE employs integer strings as genotypic representation and each integer value (called codon) can have a decimal value in the range of $[0, 255]$. Then GE performs the mapping process on the given integer string by means of the modulo operation, by which a non-terminal is replaced with an alternative. In other words, GE takes the remainder that a codon is dived by the number in RHS. This process is repeated from the leftmost non-terminal symbol until the phenotype contains no non-terminal. Furthermore, there are two exceptions that may occur in this process. In this example, GE makes use of six integer codons to generate the phenotype and then the remaining part is discarded. If there are no codons left to read, GE reuses the given genotype; the given integer string is read again from the leftmost symbol (called wrapping). Note that GE assigns the worst possible fitness value when the mapping process exceeds the pre-defined number of wrapping.

In Fig. 1, let us assume that we have a target function of $X * X$. It is clear that GE has to choose promising options in Steps 6 and 7 (i.e., operator $*$ and variable X). However, it is hard to guarantee a proper rule selection in GE because of the nature of mapping process, which uses the modulo function to choose an RHS. In other words, all the symbols have an equal chance to be chosen and this characteristic causes to spend more codons. In other words, it can be beneficial to search for the optimal solution(s) if different but proper chances are given to RHSs and their conditional dependencies are further considered. Note that the overall procedures of the proposed conditional dependency based PMBGE (cdPMBGE) are described in Algorithm 1.

2.1 PCFG and Mapping Process

We employ real-coded codons to deal with PCFG since the production rules have their own probabilities related to the selection of RHS in the mapping process. Each codon also has a real number in the range of $[0, 1]$. In the initial stage, the probability value of each RHS is identical at every non-terminal. The proposed algorithm then changes the probability values by means of the estimation of distribution of the given grammars. In other words, the mapping process is performed on the basis of the probabilities of production rules. To be exact, the given codons are read from the leftmost one and the values of codons (i.e., probabilities of RHSs) are used to determine the selection of production rules. Then the remaining part of genotype is discarded when the mapping process is completed before reaching the end of codons. Moreover, the conventional wrapping operation is modified in such a manner that necessary codons are randomly regenerated rather than the reuse of codons; this feature is beneficial to maintain the diversity of population. Figure 2 shows an example of PCFG and mapping process, which will be used for the royal tree problem in Sect. 3.

2.2 Estimation of Distribution of Grammars

The proposed algorithm generates a complete program by means of the probabilistic modeling of the given PCFG; the given grammar set is evolved towards a suitable distribution on its production rules. Firstly, the algorithm starts with an initial (probabilistic) model of PCFG; an equal probability is assigned to all replacement symbols for each non-terminal. Then the mapping process is performed to transform real-coded strings into program codes. After that, the algorithm evaluates the population (i.e., phenotypes) that is generated by the mapping process. Each individual has its own fitness value that is computed by the given fitness function (called objective function) and the value indicates how an individual is close to the optimal solution. After choosing promising individuals by selection operator, the distribution of grammars is estimated by counting the number of occurrences of each production rule in the selected individuals. In addition, conditional dependencies between production rules are considered to incorporate the knowledge of problem structure into the individual (i.e., program) generation. As a result, the probabilistic model is utilized to properly choose production rules in the mapping process.

In the estimation process, a tree structure (called Conditional Dependency Tree: CDT) is employed, by which mutual relationships between production rules are represented. Meanwhile, the tree structure can also deal with production rules that have no dependency with respect to another non-terminal. Figure 3 depicts an example of CDT on the grammar set in Fig. 2. Note that the nodes (Tree$_A$, Tree$_B$, and Tree$_C$) indicate the RHSs of non-terminal <intnode>. In addition, RHS$_{12}$ and RHS$_{23}$ represent the second RHS of the first non-terminal and the third RHS of the second non-terminal, respectively. For instance, a non-terminal <intnode> can be replaced with three alternatives (children nodes), as shown in Fig. 3(a). Then the algorithm assigns a new (updated) probability value to each RHS, as follows:

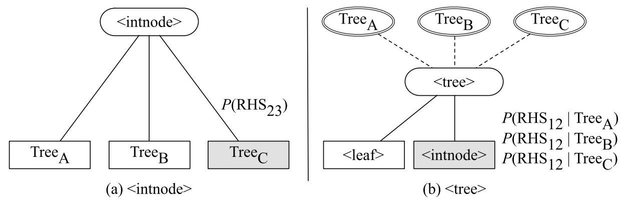

Fig. 3 An example of conditional dependency tree

$$Prob.(RHS_{ij}) = \frac{freq(RHS_{ij})}{\sum_{i=1}^{N} freq(RHS_i)} \quad (1)$$

where RHS_{ij} is the jth RHS of the ith non-terminal, $freq(RHS_{ij})$ is the number of occurrences of RHS_{ij}, N is the number of production rules of the ith non-terminal, and $freq(RHS_i)$ denotes the frequency of all instances of the ith non-terminal in the mapping process.

Furthermore, the non-terminal <tree> has two child RHSs and three parent nodes (see Fig. 3(b)). It denotes that the mapping of that symbol <tree> is influenced by the parent rule. In this sense, three different probabilities exist at the second RHS (i.e., <intnode>) where the non-terminal <tree> comes from. Thus, a different but more promising probability can be assigned to each RHS in accordance with its parent rule.

2.3 Program Generation from Estimated Distribution

In this work, EDA mechanisms are utilized as a search engine; conventional operators such as crossover and mutation are replaced with the sampling of the estimated distribution. It is important to update the probabilities of production rules (i.e., building a probabilistic model) since a set of offspring (i.e., new individuals) is generated by sampling the distribution. In the proposed approach, this process is performed by the mapping process along with the updated probability distribution in conjunction with the (discovered) conditional dependency tree. Finally, the replacement operation is carried out at the end of generation.

3. Experimental Results

In order to prove the effectiveness of our approach, the proposed algorithm was compared with three existing methods; conventional GE (CGE) [1], multi-chromosomal GE (MCGE) [2], and univariate (probabilistic) model building GE (UMBGE) [3]. Moreover, three benchmark problems and one real-world problem were considered [3], [5], [6]. For all test cases, we used the population size - 1000, the number of codons - 20, and the maximum number of wrapping - 20. Apart from these parameters, the pair-wise tournament selection and the elitist replacement were employed, and 10^5 evaluations were taken as the termination condition. Only for CGE and MCGE, the probabilities of crossover and mutation were set to 0.9 and 0.1, respectively.

The three benchmark problems include two easy problems (i.e., symbolic regression and even-5-parity) and one

```
<expr> ::= <expr> <op> <expr>        <op> ::= + | − | * | /
         | (<expr> <op> <expr>)      <pre-op> ::= Sin | Cos | Exp | Log
         | <pre-op> (<expr>)         <var> ::= x1 | x2 | x3 | x4 | x5 | x6 | x7
         | <var>
```

Fig. 4 The grammar for intention recognition problem

hard problem (i.e., royal tree). First, the symbolic regression problem is to find a symbolic-form mathematical expression, which accurately represents the given set of input and output values. On the target function of $f(X) = X^4 + X^3 + X^2 + X$, 20 points were chosen in the range of $[-1, 1]$ as input data, and the fitness value is computed by the root mean square error of sum. Second, the even-5-parity problem is to find a boolean expression for random strings of 0's and 1's, which returns 'true' for the even number of '1' in the five bits and 'false' for other cases. The fitness value is obtained by the number of incorrect outputs in 32 combinations of 5-bit string. Meanwhile, details on the grammars defined for the above two problems can be found in [3]. Third, the royal tree problem is to construct a perfect tree at the given level, which consists of different perfect trees at lower levels. For instance, the perfect tree at level C has a symbol 'C' as a root node with three perfect trees at level B as children. As the fitness value, the score on the obtained tree is computed recursively from the root node. As mentioned earlier, the grammar set defined on this problem has been given in Fig. 2. More details on the scoring metric can be found in [5].

As a real-world application, an intention recognition problem was tested. The task is to find an expression that classifies the given brain signal data into an appropriate intention pattern. we used the data set acquired in [6], which contains 280 samples and 7 features that are comprised of real numbers in the range of $[0, 1]$. More details can be found in [6]. Moreover, the grammar set for this problem is given in Fig. 4, in which RHSs of non-terminal <var> indicate the features of data set. The fitness value is computed by the difference between the expression output and the actual pattern.

Table 1 shows the experimental results on the three benchmark problems. The results were average over 100 runs. For the first and second benchmark problems (i.e., symbolic regression and even-5-parity), the existing algorithms except for CGE performed better than cdPMBGE. It denotes that the knowledge on conditional dependency (of production rules) is not so effective on the problems of simple structure. It is nothing to worry about because the three algorithms (i.e., cdPMBGE, UMBGE, and MCGE) seem to be statistically comparable with each other (when considering the standard deviations). Meanwhile, our cdPMBGE significantly outperformed all the existing algorithms on the royal tree problem (i.e., difficult problem). It implies that the mutual relationship between production rules plays an important role in discovering the optimal solutions in the problems of complex structure. In other words, the proposed algorithm is able to handle the correlation in the form

Table 1 Experimental results for three benchmark problems: mean best fitness, cumulative frequency of success, and standard deviation.

Symbolic regression

	cdPMBGE	UMBGE	MCGE	CGE
Mean best fit.	0.12	0.08	0.40	0.37
Cum. frequency	88	97	17	11
Std. deviation	0.09	0.07	0.12	0.14

Even-5-parity

	cdPMBGE	UMBGE	MCGE	CGE
Mean best fit.	6.30	5.03	14.54	14.07
Cum. frequency	39	81	4	11
Std. deviation	3.72	4.32	2.51	5.05

Royal Tree (Level C)

	cdPMBGE	UMBGE	MCGE	CGE
Mean best fit.	28.93	301.60	294.33	321.00
Cum. frequency	86	-	-	-
Std. deviation	1.86	21.95	19.92	5.13

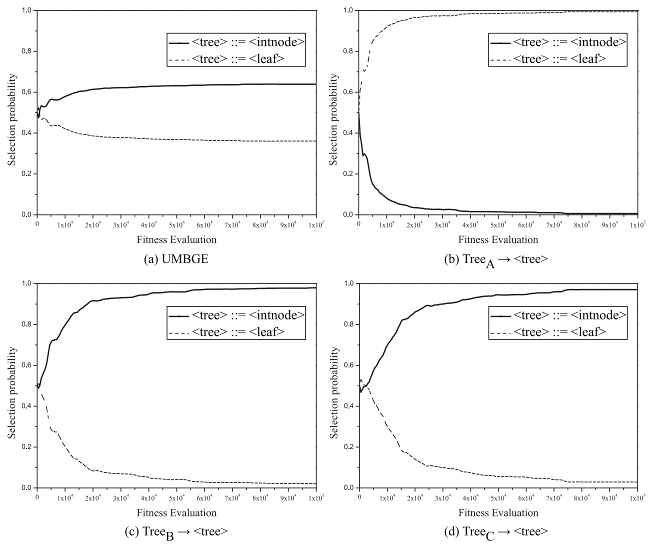

Fig. 5 Trace of selection probabilities of non-terminal <tree> on the royal tree problem: (a) UMBGE, (b)–(d) cdPMBGE

of conditional dependency tree by involving a specific non-terminal in the mapping process of another non-terminal.

Figure 5 depicts the trend of selection probabilities of the non-terminal <tree> on the royal tree problem. It is seen that cdPMBGE dealt with each symbol individually in accordance with its parent rule, while UMBGE handles all of the symbols by the same distribution. Meanwhile, it is known that the conventional algorithms (i.e., MCGE and CGE) employ the modulo operation to select an RHS; all RHSs are handled with the same probability values. It denotes that our approach is beneficial to search for the perfect tree since the selection probability of each RHS of <tree> is adjusted in the course of evolution.

For the intention recognition problem, the experimental results are shown in Table 2. The results were averaged over 50 runs and the Welch t-test (two-tailed) was performed to determine statistical significance. It is observed that our cdPMBGE outperformed CGE with regard to classification

Table 2 Experimental results for intention recognition problem: mean average fitness, standard deviation, and classification accuracy.

	cdPMBGE	UMBGE	MCGE	CGE
Mean average fit.	6.94	7.35	7.74	8.04
Std. deviation	3.26	3.50	3.28	3.61
Classification (%)	75	74	72	71
t-test case	Our *vs.* UMBGE	Our *vs.* MCGE	Our *vs.* CGE	
p-value	0.051	0.044	0.004†	

† It stands for statistical difference ($p < 0.005$).

accuracy; this was supported by the statistical test. Despite the difference not being statistically significant, it appears that cdPMBGE achieved a bit better performance than UMBGE and MCGE. This denotes that the proposed approach can generate a more complete program (i.e., accurate classifier) for the brain signal classification problem.

4. Conclusion

In this paper, a new approach to probabilistic model based grammatical evolution that considers conditional dependency of given grammars to evolve candidate solutions was proposed. To achieve this goal, the strengths of GE and the concept of EDA were combined together. Instead of conventional genetic operators, probabilistic context-free grammar was employed in the probabilistic modeling and sampling process. Furthermore, the mapping process was modified to generate a complete program on the basis of the distribution of production rules in the given grammars. Experiments were carried out on several benchmark problems, and the intention recognition from brain signal data was tackled as a real-world problem. The experimental results demonstrated the effectiveness of our approach on difficult and complex problems.

Acknowledgments

This research was supported by X-Project funded by the Ministry of Science, ICT & Future Planning (NRF-2015R1A2A1A16074929).

References

[1] M. O'Neill and C. Ryan, "Grammatical evolution," IEEE Trans. Evol. Comput., vol.5, no.4, pp.349–358, 2001.

[2] A. Hara, T. Yamaguchi, T. Ichimura, and T. Takahama, "Multi-chromosomal grammatical evolution," Intl. Workshop on Computational Intelligence & Applications, pp.37–42, 2008.

[3] H.T. Kim and C.W. Ahn, "A new grammatical evolution based on probabilistic context-free grammar," Symposium on Intelligent and Evolutionary Systems, pp.1–12, Springer, 2015.

[4] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," Swarm Evol Comput, vol.1, no.3, pp.111–128, 2011.

[5] W.F. Punch, "How effective are multiple populations in genetic programming," Genetic Programming, vol.98, pp.308–313, 1998.

[6] J.H. Lee, J.R. Anaraki, C.W. Ahn, and J. An, "Efficient classification system based on fuzzy–rough feature selection and multitree genetic programming for intension pattern recognition using brain signal," Expert Syst. Appl., vol.42, no.3, pp.1644–1651, 2015.