Parallel Implementation of Grammatical Evolution

***E. Kita[1,2], Y. Lu[1], H. Sugiura[1] and Y. Wakita[1]**

[1]Department of Complex Science, Graduate School of Information Science, Nagoya University
Nagoya, JAPAN
[2]Department of Computational Science, Graduate School of System Informatics, Kobe University
Kobe, JAPAN

*Corresponding author: kita@is.nagoya-u.ac.jp

Abstract

Grammatical Evolution (GE) is one of the evolutionary computations, which determine function or program or program fragment satisfying the design objective, like Genetic Programming (GP). The interesting feature of GE is to define the translation rule from the genotype (bit-string) to the phenotype (function or program) in advance. The population of individuals (bit-strings) is evolved toward better individual by using the translation rule and Genetic Algorithm (GA) search process.

The aim of this study is to discuss the effectiveness of parallel implementation for GE. The parallel implementation is based on simple island model. The whole population is substituted into sub-populations. The evolution process is performed individually in subpopulations and some individuals are exchanged between subpopulations in any interval. Exchange of individuals is called as migration. Better individual migration and randomly selected individual migration are compared.

The symbolic regression problem is considered as a numerical example. The results show that, in random migration, longer migration interval is better for larger sub-population size and shorter migration interval is better for smaller sub-population size and that, in better individual migration, longer migration interval and larger sub-population size are better.

Keywords: Grammatical Evolution, Parallel Implementation, Island Model, Symbolic Regression Problem.

Introduction

Evolutionary computations are algorithms based on the evolutionary process of living organism. Genetic Algorithm (GA) and Genetic Program (GP) are very popular algorithms in this field [1,2]. The aim of GA is to find the solution of the optimization problem. Potential solutions are represented by individuals as bit-strings. The population of individuals evolves to a better potential solution by genetic operators such as selection, crossover, mutation, and so on. Although GP comes from GA, their aims are different. GP is designed for finding the function, the program or the program fragment satisfying the design objective. The potential solutions, which are represented as the individuals in tree structure, evolve toward better solution by genetic operators.

GP has two difficulties. Firstly, the genetic operators are very complicated and the effect for the search process is not obvious. Secondly, during the GP search process, genetic operators often generate individuals which lead to invalid function or invalid program or invalid program fragment. For overcoming these difficulties, Grammatical Evolution (GE) was presented [3,4]. The interesting feature of GE is to define the translation rule from genotype (bit-string) to phenotype (function or program) in advance. The translation rule is written in Backus Naur Form (BNF). Once the valid translation rule is given, GE can generate the genotypes which lead to valid phenotypes. The search process of GE is as follows. Initial population of individuals is defined by randomly generated bit-strings. Genotypes are translated into the phonotype according to the translation rule and the fitness is estimated. According to the fitness, the population of individuals is evolved toward better solutions by the genetic operators.

Except for the use of the translation rule, the GE search process is very similar to simple GA. Therefore, the use of the improved GA algorithm can improve the search process of the original GE. The aim of this paper is to use parallel GA for GE. The parallel implementation of GA is studied widely [5]. The distributed Genetic Algorithm (DGA) based on Island model is employed in this study [6]. In DGA, the population is divided into sub-populations. GA is applied for each sub-population and then, individuals migrate from one sub-population to the other one at regular interval. It is reported that DGA based on island model can find better solution than traditional GA using single population. The algorithm is applied for symbolic regression problem in order to discuss the search performance.

Table 1: Example of translation rule

(A)	<expr> ::= <expr><op><expr> | <var>	(A0) (A1)
(B)	<op> ::= + | - | * | /	(B0) (B1) (B2) (B3)
(C)	<var> ::= X | Y | Z	(C0) (C1) (C2)

Grammatical Evolution

Algorithm

The algorithm of the original GE is summarized as follows.

1. Translation rule is defined in Backus Naur Form (BNF).
2. Initial population is defined by randomly generated bit-strings.
3. Genotypes (bit-strings) are translated into phenotypes (function or program) according to the translation rule.
4. Phenotype fitness is estimated.
5. Population is updated by genetic operators such as selection, crossover and mutation.
6. If the convergence criterion is satisfied, the process is terminated. Otherwise, process goes to step 3.

Translation from Genotype to Phenotype

We would like to explain the translation from genotype from phenotype. The translation rule in BNF syntax is shown in Table 1. It is shown from this table that the symbol <expr> has two candidate symbols <expr><op><expr> and <var> and that the symbol <op> has four candidate symbols +, -, * and / and that the symbol <var> has three candidate symbols X, Y and Z. The symbols +, -, * and / denote the four arithmetic operators and the symbols X, Y and Z are variables. Since the symbol <expr><op><expr> and <var> should be replaced again, they are called as recursive symbols. The symbols +, -, *, /, X, Y and Z are called as terminal rules because they are not replaced any more.

When the start symbol is <expr> and the genotype is given as the binary 010001111101101110, the genotype is translated according to Table 1 as follows (Table 2).

1. The binary number 010001111101101110 is translated into the decimal number every 3bits as follows.

   010001111101101110 → 2 1 7 5 5 6

2. The symbol <expr> has two candidate rules and the first decimal number is 2. The remainder of the decimal number 2 with respect to the candidate symbol number 2 is 0. Then, the symbol <expr> is replaced with <expr><op><expr>.
   <expr> → <expr><op><expr>.
3. Next replaced symbol is the leftmost recursive symbol <expr>.
4. The symbol <expr> has two candidate rules and the next decimal number is 1. The remainder of the decimal number 1 with respect to the candidate symbol number 2 is 1. Then, the symbol <expr> is replaced with <var>;
   <expr><op><expr> → <var><op><expr>.
5. According to the similar process, the function Y+X is obtained from the binary number 0100011111101101110.

Table 2: Symbol replacement process

Decimal	Remainder	Target symbol	Selected symbol	Symbol after replacement
Start				<expr>
2	0	<expr>	<expr><op><expr>	<expr><op><expr>
1	1	<expr>	<var>	<var><op><expr>
7	1	<var>	Y	Y<op><expr>
5	4	<op>	+	Y+<expr>
5	1	<expr>	<var>	Y+<var>
6	0	<var>	X	Y+X

Parallel Grammatical Evolution

Algorithm

In the present algorithm, the parallel implementation of Grammatical Evolution is performed according to Genetic Algorithm based on the island model. The whole population of the individuals is divided into sub-populations and then, the original GE is performed at each sub-population. The present algorithm is summarized as follows.

1. Translation rule is defined in Backus Naur Form (BNF).
2. Initial sub-populations are defined by randomly generated bit-strings.
3. Genotypes (bit-strings) are translated into phenotypes (function or program) according to the translation rule.
4. Individual fitness is estimated.
5. Sub-populations are updated by genetic operators such as selection, crossover and mutation.
6. If the convergence criterion is satisfied, the process is terminated. Otherwise, the process goes to next step.
7. Individuals are migrated from one sub-population to the other one at any interval.
8. Process goes to step 3.

Migration

Migration operator exchanges the individuals between the sub-populations. In this study, two migration operators are compared.

The individuals to be migrated are selected as follows.

1. Random migration.
   Immigrant individuals are selected randomly from the sub-populations.
2. Better individual migration
   Immigrant individuals are selected from the sub-populations according to the descending order of the fitness value.

The migration frequency is given by the migration interval. The number of the migrated individuals is given as the product of the number of individuals and the migration rate. The migration topology is fixed.

Table 3: Translation rule

(A)	<expr> ::= <expr><op><expr>	(A0)
	| <var>	(A1)
(B)	<op> ::= +	(B0)
	| -	(B1)
	| *	(B2)
	| /	(B3)
(C)	<var> ::= X	(C0)
	| Y	(C1)
	| Z	(C2)
(D)	<num>::= 1	(D1)
	| 2	(D2)
	| 3	(D3)
	| 4	(D4)
	| 5	(D5)
	| 6	(D6)
	| 7	(D7)
	| 8	(D8)
	| 9	(D9)

Table 4: Simulation parameters

Length of individual	800
Radix conversion bit-size	8 bit
Tournament size	3
Elite size	1
Crossover rate ϕ_c	0.9
Mutation rate ϕ_m	0.03
Number of sub-populations n_s	2, 5, 10
Migration rate η_r	0.1, 0.2, 0.5
Migration interval η_i	2, 4, 8, 16

Numerical Example

Symbolic Regression Problem

Symbolic regression problem is to find the function which can represent accurately the given data set; $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$. The real function is given as follows.

$$f(x) = 4x^2 - 12x + 9 \tag{1}$$

The variable x is given as $\{x_1, x_2, x_{50}\} = \{-2.5, -2.4, \cdots, -0.1, 0.1, \cdots, 2.4, 2.5\}$.

The fitness is estimated by the average least square error as follows.

$$E = \frac{1}{50}\sqrt{\sum_{i=1}^{50}\left\{f(x_i) - \bar{f}(x_i)\right\}^2} \tag{2}$$

The translation rule is shown in Table 3. The start symbol is <expr>. The tournament selection with tournament size 3, one-point crossover operators and elitist strategy are employed.

Random Migration

Simulation parameters are shown in Table 4. Total number of individuals is 200. Maximum generation is 100. Simulations are performed 100 times. Success rates are shown Tables 5, 6 and 7. The success rate denotes, among 100 simulations, the percentage of simulations at which the exact function can be found.

In case of the migration rate $\eta_r = 0.1$, the fastest convergence is observed at the number of sub-populations $n_s = 5$ and the migration interval $\eta_i = 8$. Second fastest convergence is at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 8$ or at the number of sub-populations $n_s = 10$ and the migration interval $\eta_i = 2$. In case of the migration rate $\eta_r = 0.2$, the fastest convergence is observed at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 4$. Second fastest convergence is at the number of sub-populations $n_s = 5$ and the migration interval $\eta_i = 8$. In case of the migration rate $\eta_r = 0.5$, the fastest convergence is observed at the number of sub-populations $n_s = 10$ and the migration interval $\eta_i = 2$. Second fastest convergence is at the number of sub-populations $n_s = 5$ and the migration interval $\eta_i = 8$. It is concluded that longer migration interval is better for larger sub-population size and shorter migration interval is better for smaller sub-population size.

Table 5: Comparison of convergence speed (Random migration ; migration rate $\eta_r = 0.1$)

	$\boldsymbol{\eta_i = 2}$	$\boldsymbol{\eta_i = 4}$	$\boldsymbol{\eta_i = 8}$	$\boldsymbol{\eta_i = 16}$
$\boldsymbol{n_s = 2}$	72	73	75	70
$\boldsymbol{n_s = 5}$	68	73	76	72
$\boldsymbol{n_s = 10}$	75	66	65	60

Table 6: Comparison of convergence speed (Random migration ; migration rate $\eta_r = 0.2$)

	$\boldsymbol{\eta_i = 2}$	$\boldsymbol{\eta_i = 4}$	$\boldsymbol{\eta_i = 8}$	$\boldsymbol{\eta_i = 16}$
$\boldsymbol{n_s = 2}$	69	82	75	70
$\boldsymbol{n_s = 5}$	73	79	81	65
$\boldsymbol{n_s = 10}$	75	74	69	55

Table 7: Comparison of convergence speed (Random migration ; migration rate $\eta_r = 0.5$)

	$\boldsymbol{\eta_i = 2}$	$\boldsymbol{\eta_i = 4}$	$\boldsymbol{\eta_i = 8}$	$\boldsymbol{\eta_i = 16}$
$\boldsymbol{n_s = 2}$	70	70	61	63
$\boldsymbol{n_s = 5}$	73	74	81	74
$\boldsymbol{n_s = 10}$	82	72	67	52

Better Individual Migration

Simulation parameters are shown in Table 4. Total number of individuals is 200. Maximum generation at each simulation is 100. Simulations are performed 100 times. Success rates are shown Table 8, 9 and 10.

At the migration rate $\eta_r = 0.1$, the fastest convergence is observed at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 4$. Second fastest convergence is at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 16$. In case of the migration rate $\eta_r = 0.2$, the fastest convergence is observed at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 8$. Second fastest convergence is at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 4$ or at the number of sub-populations $n_s = 5$ and the migration

interval $\eta_i = 16$. In case of the migration rate $\eta_r = 0.5$, the fastest convergence is observed at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 16$. Second fastest convergence is at the number of sub-populations $n_s = 2$ and the migration interval $\eta_i = 4$. It is concluded that longer migration interval and larger sub-population size are better.

Table 8: Comparison of convergence speed (Random migration; migration rate $\eta_r = 0.1$)

	$\boldsymbol{\eta_i = 2}$	$\boldsymbol{\eta_i = 4}$	$\boldsymbol{\eta_i = 8}$	$\boldsymbol{\eta_i = 16}$
$\boldsymbol{n_s = 2}$	63	72	65	70
$\boldsymbol{n_s = 5}$	62	60	62	61
$\boldsymbol{n_s = 10}$	37	49	42	50

Table 9: Comparison of convergence speed (Random migration; migration rate $\eta_r = 0.2$)

	$\boldsymbol{\eta_i = 2}$	$\boldsymbol{\eta_i = 4}$	$\boldsymbol{\eta_i = 8}$	$\boldsymbol{\eta_i = 16}$
$\boldsymbol{n_s = 2}$	54	67	68	61
$\boldsymbol{n_s = 5}$	49	45	59	67
$\boldsymbol{n_s = 10}$	37	46	46	39

Table 10: Comparison of convergence speed (Random migration; migration rate $\eta_r = 0.5$)

	$\boldsymbol{\eta_i = 2}$	$\boldsymbol{\eta_i = 4}$	$\boldsymbol{\eta_i = 8}$	$\boldsymbol{\eta_i = 16}$
$\boldsymbol{n_s = 2}$	56	65	60	73
$\boldsymbol{n_s = 5}$	44	57	58	55
$\boldsymbol{n_s = 10}$	26	35	44	38

Conclusion

Parallel implementation of Grammatical Evolution based was presented in this study. The algorithm is based on island model. The present algorithm was applied for the symbolic regression problem. In random migration, it is concluded that longer migration interval is better for larger sub-population size and shorter migration interval is better for smaller sub-population size. In better individual migration, longer migration interval and larger sub-population size are better. In the future, we would like to apply the present algorithm to industrial application problems.

References

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

[2] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press. 1992.

[3] C. Ryan, J. J. Collins, M. O'Neill. Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Proceedings of 1st European Workshop on Genetic Programming*, pp.83-95, Springer, 1998.

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

[5] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms*, Springer, 2000.

[6] R. Tanese. Distributed Genetic Algorithms. *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp.434-439, Morgan Kaufmann, 1989.