High-speed Generation of Logic Function to Identify Exon-Intron Boundaries by Parallel GP

Kunihito YAMAMORI[1] Yuji FUJITA[2] Ikuo YOSHIHARA[3]
Masaru AIKAWA[4]

Abstract

Genetic Programming (GP) is applied to various field. However, it is a problem that GP needs a lot of execution time for large scale problems such as genome informatics. In this paper, we focus on reducing computation time of GP. For this objective, we parallelize GP by island model on a personal computer (PC) cluster system to identify exon-intron boundaries in DNA sequences which is example of application. In addition, we described individuals as an one dimentional matrix to reduce the time of migration which is a process of island model. The paralellized GP achieved linear speedup ratio on each number of PC. Moreover, we show quality of solution to change on each number of PC.

Key Words:
Genetic Programming, Prallel Computing, Cluster System, Island Model, Matrix Representation Individual

1 Introduction

Genetic Programming (GP) is applied to many real-world problems such as time series prediction, pattern recognition and so on[1]. It is an advantage that GP can produce various models represented as individuals because they are described by structural representation such as the tree structure. These individuals are candidate of solution, and they are evolved through some genetic operations to find the optimal solution. For search better solution by GP, GP need many individuals and generations for accurate solution lead unacceptable computation time. Parallel processing is one of the effective way for the problem which need many computation time. Many parallel computers have been utilized in recent years. Nowaday, Personal Computer (PC) cluster consisting of multiple PCs and network attract attention. Many parallel GP with island model on PC cluster have been implemented and evaluated[2]. However, overhead of migration in the island model disturbs high efficient processing on PC cluster with distributed memory. We parallelize GP on PC cluster to reduce execution time by representing individuals as one dimensional matrix to reduce the overhead of migration, then it is evaluated. This paper is organized as follows; we briefly introduce the GP in Section 2. Section 3 shows the system environment of the PC cluster, then explains how to parallelize GP. Simulation results and discussions are described in Section 4 and Section 5 concludes this paper.

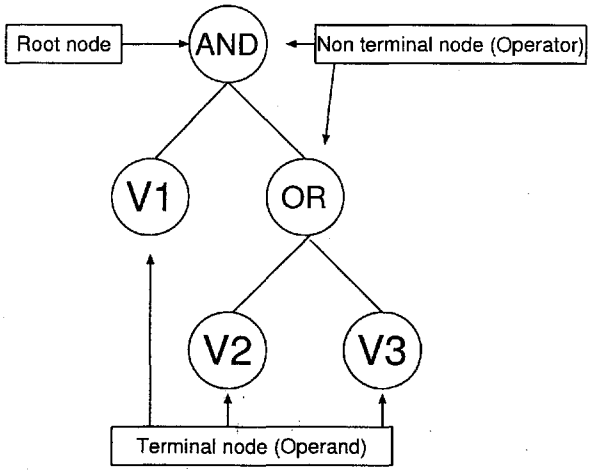

Fig. 1 An example of individual in GP.

2 Genetic Programming

2.1 Outline of Genetic Programming

Genetic Algorithm (GA) is one of the evolutionary computing based on biological evolution. In GA, an in-

[1] As.Prof., Dept. Comp. Sci. and Sys. Eng., Univ. MIYAZAKI
[2] Undergraduate Student, Dept. of Comp. Sci. and Sys. Eng., Univ. MIYAZAKI
[3] Prof., Dept. of Comp. Sci. and Sys. Eng., Univ. MIYAZAKI
[4] Tech.Staff, Dept. of Comp. Sci. and Sys. Eng., Univ. MIYAZAKI

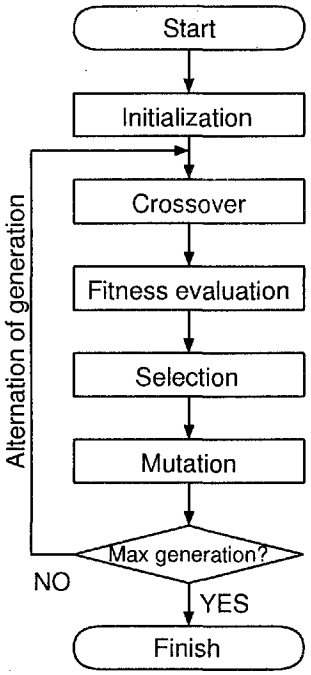

Fig. 2 Algorithm of Genetic Programming.

Table. 1 Using operator in GP.

number of argument	Logic function
1	NOT
2	AND OR XOR NAND NOR XNOR
3	M(2/3) (Majority decision) NM(2/3) (NOT M(2/3))

dividual represents a solution of the problem.

Genetic Programming (GP) is an extension of GA. While GA represents an individual by a one dimensional matrix, GP represents an individual by a tree structure as shown in Figure 1. Individual described by tree structure can represent equations, models and relations among article. Processes of GP shown by Figure 2 are the almost same as those of GA except operation of crossover and mutation operations. The crossover in GP means exchange of subtree since the individual of GP represent the structural model. The mutation is defined by changing the operator in the non-terminal node to the other operator with the same number of arguments.

Figure 3 and Figure 4 show the example of the crossover and mutation operation in GP, respectively.

In this paper, GP generates a logic function to identify the exon-intron boundaries in DNA sequences. The logic function is represented a tree structure as shown in Figure 1. An individual consists of a root node, non-terminal nodes and terminal nodes. Each non-terminal node has a operator which is kinds of logic function as shown Table 1. In Table 1, M(2/3) is majority decision of three argument. NM(2/3) is inverter of the M(2/3). Terminal nodes

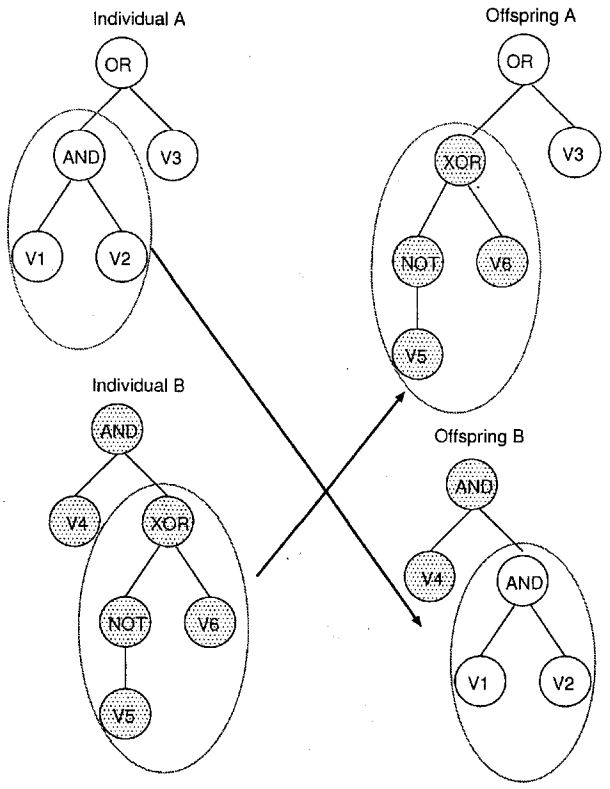

Fig. 3 Crossover operation in GP.

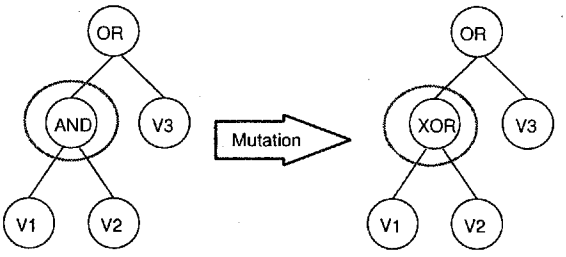

Fig. 4 Mutation operation in GP.

are operand whose value is 0 or 1.

2.2 Boundary between exon and intron

DNA sequences consist of four kinds of bases, Adenine (A), Cytosine (C), Guanine (G) and Thymine(T). DNA can be divided into two parts; the exon and the intron. An exon area hold the information to synthesis protein. The intron does not have any information to synthesis protein. The exon and the intron alternatively appear in the DNA sequences. The intron has a rule which begins from the "AG" and finishes at the "GT". Although many "AG" and "GT" exist DNA sequences. So we need to decide whether "AG" and "GT" are a boundary or not in every

Table. 2 Encoding of bases.

Base	Encoding
A	(0 0 0 1)
C	(0 0 1 0)
G	(0 1 0 0)
T	(1 0 0 0)

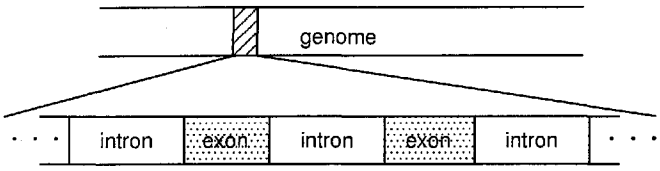

Fig. 5 The structure of DNA.

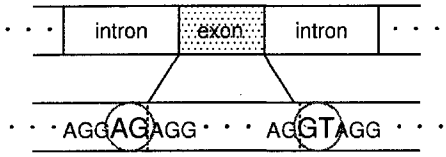

Fig. 6 Rule of GT-AG.

"AG" and "GT".

In this paper, we try to make a logic function by GP that identifies the "AG"s and the "GT"s are the boundary or not. The logic function generated by GP outputs 1 if the focused "AG" or "GT" is a boundary. In other cases, it outputs 0[3]. For this objective, each base is encoded to the binary as shown in Table 2. The binary inputs to the logic function are made from the bases before and after the focused "AG" and "GT".

The bases near by "AG" and "GT" as shown in Figure 7 and Figure 8 are respectively used as the sample to make a model.

2.3 Definition Fitness

The fitness is defined by Equation (1).

$$fitness = \sum_{i=1}^{T} X_i + \frac{1.0}{m}, \quad (1)$$

$$Xi = \begin{cases} 1 & \bar{Y}_i = Y_i, \\ 0 & \bar{Y}_i \neq Y_i. \end{cases}$$

In equation (1), T denotes the number of sample data, m denotes the number of operator and operand. Therefore, $fitness$ become larger when a model can correctly identify more and more boundaries, and a model has fewer and fewer operator and operand. GP needs long computation times; In particular the fitness evaluation dominates the total computation time[2]. To reduce the computation time, we parallelize the GP by the island model on PC cluster.

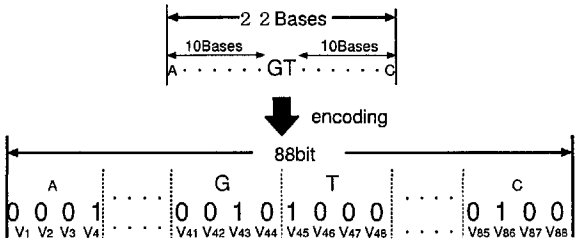

Fig. 7 Encode sequence near by "GT"

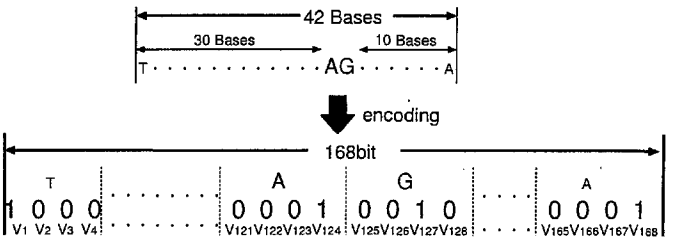

Fig. 8 Encode sequence near by "AG"

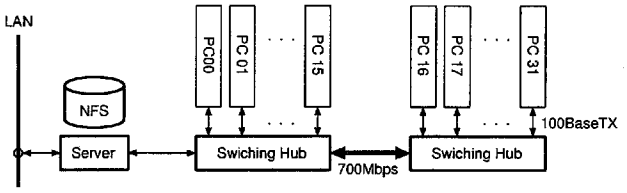

Fig. 9 Architecture of PC cluster system.

3 Parallel Genetic Programming

3.1 PC cluster

PC cluster operates on multiple PC with a TCP/IP network. PC cluster can increase in throughput and extend the a memory space by using multiple PC.

However, PC cluster has relatively large communication overhead among PCs. Therefore the number and the amount of messages have to be reduced as few as possible for efficient use of PC cluster.

In this paper, we employ a beowulf type PC cluster system. PC equip a Pentium III 933MHz processor and 256MB main memory. PCs are connected by 100Base-T Ethernet and TCP/IP network. The system architecture is shown in Figure 9. Each PC shares the hard disk on the server by Network File System (NFS).

3.2 Message Passing Interface

In this paper, we used Message Passing Interface (MPI) for the communication among PCs. MPI is a library which are operated by C and FORTRAN language. MPI provides easy communication among PCs[4].

3.3 Island Model

We applied island model to parallelize GP. The island model are called Distributed GA (DGA) in evolutionary computation. In the island model, the population is divided into some sub-populations. Then a sub-population is assigned to a PC. PCs independently execute GP operations for the individuals in their own sub-population. Every some generations, PCs exchange some individuals each other. This exchange process is called as "Migration". Since the number of individuals for a PC is decreased and the genetic operations are executed in parallel, the computation time can be reduced. In addition,

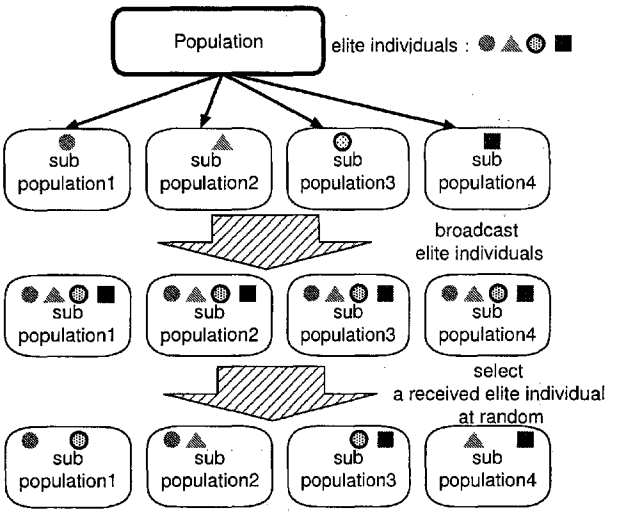

Fig. 10 Island model and process of migration.

the variety of the population is also kept by the migration process.

Figure 10 shows an example of island model using four PCs and process of migration. The detail of the migration is as follows;

1. A population is divided into four sub-populations, and each sub-population is assigned to a PC.
2. The genetic operations are executed in parallel on each PC, then the elite that has the highest fitness is selected.
3. The elite is broadcasted for all the sub-populations.
4. Only one elite is randomly selected from the broadcasted elites, then it joins in the sub-population.

3.4 One Dimensional Matrix of Individual

In C language we can represent tree structure easily by using of address pointer. However PC cluster is a distributed memory system, so we have to exchange the data not the address pointer but the actual value at the communication among PCs. From this restriction, an individual with a tree structure is represented by one dimensional matrix. The matrix representation enables to directly exchange the individuals among PCs without any additional processes. There is two manners for the representation of the tree structure by the one dimensional matrix. The one is depth priority, and the other is width priority. In the width priority manner, the nodes in a subtrees are placed on the distributed position of the individual. These distributed subtrees are hard to extract for the crossover operation. On the other hand, those are placed on a local position of the individual in the depth priority manner. This

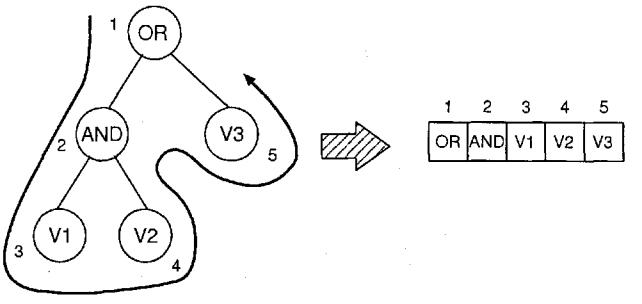

Fig. 11 A individual represented by prefix arrangement.

feature of the depth priority manner is an advantage for the crossover operation. So, we employ matrix of depth priority manner.

In the depth priority manner to represent the tree structure, there is two way to arrange the nodes; prefix and postfix arrangement[5]. In this paper, we employ the prefix arrangement.

4 Experiments and Discussions

4.1 Parameters of Simulation

The parameters to identify the exon-intron boundaries in parallel GP are as follows;

- The Number of individuals: 100

  These individuals are divided into some sub-populations, then they are assigned to the PCs and evolved for some generation.

- The Maximum number of nodes: 200

  The number of operator and operand in a individual is limited.

- The Maximum number of generations: 4,000

  Evolution process is terminated when the generation is arrived at 4,000.

- The Migration interval: 10

  The migration is executed in every 10 generations.

- The Number of simulations: 10

  10 runs for every experimental conditions from the different initial individuals.

- The Number of PCs: 1,2,4,5,10

  For load balancing, the number of PC are decided as 1,2,4,5 and 10 since the total number of individual are 100.

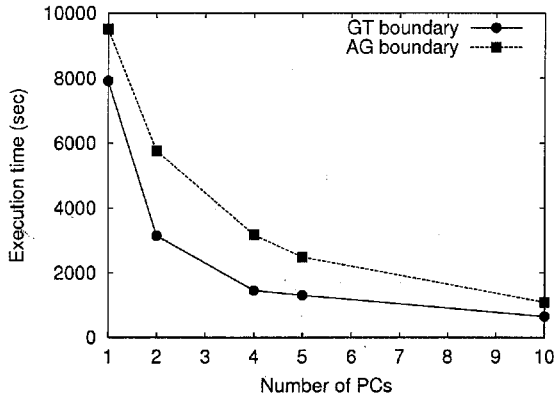

Fig. 12 Execution time for 4,000 generation on each number of PCs.

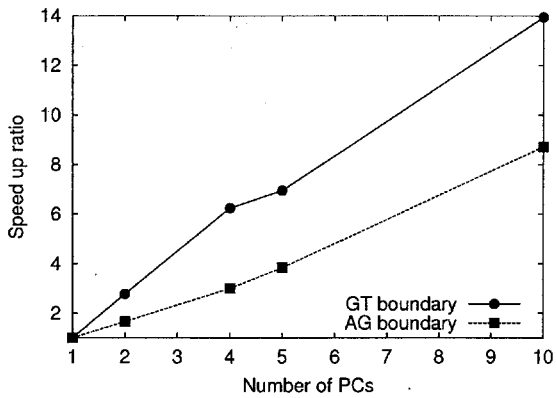

Fig. 13 Speed up ratio of parallel GP for 4,000 generation on each number of PCs.

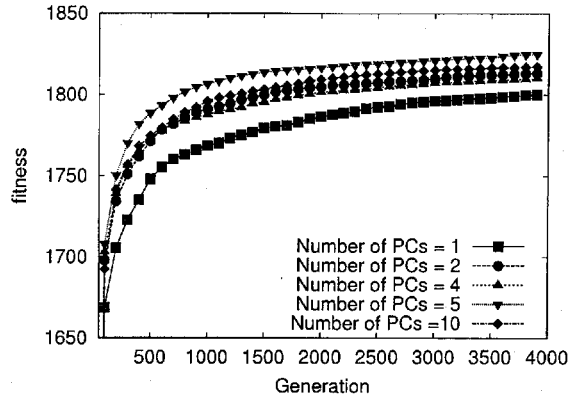

Fig. 14 Fitness of parallel GP for "GT" boundary on each number of PCs.

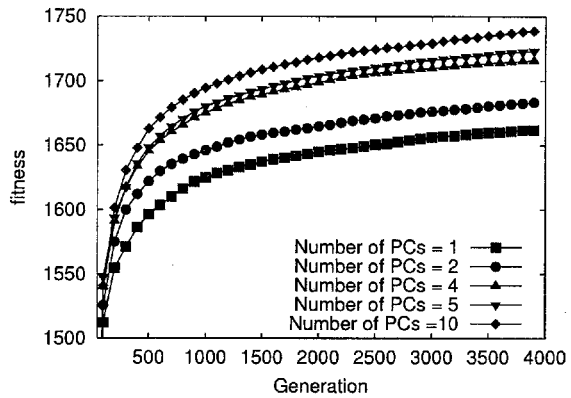

Fig. 15 Fitness of parallel GP for "AG" boundary on each number of PCs.

4.2 Experimental Results

Figure 12 shows the execution time of GP for 4,000 generation on each number of PCs. The execution time is reduced as increasing of the number of PCs. The difference between "AG" and "GT" boundary for execution time causes by the number of extracted bases near by "AG" and "GT".

Figure 13 shows the speedup ratio in each number of PCs. The parallel GP nearly achieved linear speedup.

Figure 14 and Figure 15 shows the fitness for every 100 generation. As shown in those figures, the using of the island model led higher fitness value. In the case of the "GT" boundary, the highest fitness value was achieved when the number of the sub-populations was 5. On the other hand, it was the highest fitness value for the "AG" boundary when the number of the sub-populations was 10.

Identify "GT" boundary by logic function does not need a lot of operand since simply tree of individual increase in GP operation. Increasing simple tree leads to lose various individual in sub-population. On the other hand, the case of identify "AG" boundary need a lot of operand since complexity tree of individual increased. Increasing complexity tree of individual lead to keep various individual in sub-population.

When used multiple PC, GP obtains the high fitness. Therefore, it is thought because that the population could be achieved the various evolution in each sub-population, population is hard becoming convergence. Moreover, comparing the case of 10 PCs with that of 5 PCs in Figure 14, the GP achieves higher fitness. The number of individual in sub population is too fewer as opposed to problem, individual of sub population lose the various evolution since general fitness go down.

5 Conclusions

In this paper, we focus on to reduce the execution time of GP by parallelization with island model on PC cluster system. For the benchmark, we selected the exon and intron boundary identification problem for the application problem which is very important problem in bioinfomatics. To reduce the communication overhead, we employed the one dimensional matrix representation for the

individuals which have the tree structure without the address pointer.

Parallel GP achieved linear speedup as increasing the number of PCs by using of the island model to identity both of "GT" and "AG" boundaries. In addition we showed that the fitness value was improved when the appropriate number of PCs were used. To identification of "AG" boundaries, it was needed to the complex tree structure than that for the "GT" boundaries.

Since the obtained individuals included some redundant nodes, simplification of the generated logic function is remained as a future work.

Reference

[1] Hitoshi, I. *"Genetic Programming"*. Tokyo Denki University Press,Inc, 1996.

[2] Yamamori, K., Matumoto, S., Ohta, T., Yoshihara, I. "Parallel Genetic Programming on PC Cluster SyStem", 2003. The 7th World Multiconference on Systemics, Cyvernetics and Informatics.

[3] Ohta, T., Yoshihara, I., Yamamori, K., Yasunaga, M. "GP based Method for Identifying Exon Region in DNA Sequences", 2002. Proc. The 4th Asia-Pacific Conference on Simulates Evolution and Learning.

[4] P.pacheco. *"MPI parallel programming"*. baifukan publishers,Inc, 2001.

[5] Wolfgang Banzhaf, et al. *"Genetic Programming An Introduction"*. Morgan Kaufmann Publishers,Inc, 1998.