SICE Annual Conference 2010
August 18-21, 2010, The Grand Hotel, Taipei, Taiwan

Functionally Distributed Systems Using Parallel Genetic Network Programming

Yiwen Zhang[1,2], Xianneng Li[1], Yang Yang[1], Shingo Mabu[1], *Member, IEEE,* Yi Jin[2], and Kotaro Hirasawa[1], *Member, IEEE*
[1]Graduate School of Information, Production and Systems Waseda University, Japan
(Tel: +81-933-693-5261; Email: yiwenzhang@fuji.waseda.jp, sennou@asagi.waseda.jp, yangyang@y.ruri.waseda.jp, mabu@aoni.waseda.jp, hirasawa@waseda.jp)
[2]School of Computer Engineering and Science, Shanghai University, China
(Tel: +86-21-5633-1479; Email: yijin@shu.edu.cn)

***Abstract*—Genetic Network Programming (GNP), one of the evolutionary computational methods, can generate behavior sequences of agents. In this paper, a new method named parallel GNP has been proposed and applied to functionally distributed systems consisted of several tasks. GNPs corresponding to several tasks in parallel GNP operate separately and independently but concurrently, dealing with the conflicts in task execution. Parallel GNP converges faster and has better fitness results than conventional GNP, which was shown by simulations comparing with conventional GNP on dynamic problems.**

Keywords*—*evolutionary computation; parallel Genetic Network Programming; functionally distributed systems.

I. Introduction

Many algorithms for Artificial Intelligence have been proposed on the planning of agents in order for an agent to realize a given goal in a certain environment [1]. Conventionally, how to reach the given goal is usually determined by designers. With the study of evolutionary methods and the development of parallel computing?it is, therefore?considered to be more reasonable to solve the problems by parallel evolutionary methods rather than conventional methods when the objective becomes complicated.

In this paper, a new method named parallel Genetic Network Programming has been proposed to deal with generating behavior sequences of agents. The proposed method aims to build an artificial model to achieve the functional localization of GNP. Parallel GNP consists of several GNPs, which can solve their own task, respectively. Since there are conflicts between the operations of each GNP, a probability method is also proposed to determine which task will be taken, when conflicts occur. Parallel GNP is more efficient and effective than conventional sequential GNP in terms of faster convergence and better performances.

II. GENETIC NETWORK PROGRAMMING (GNP)

Genetic Network Programming (GNP) is an evolutionary algorithm with graph structures [2]. GNP is composed by a large number of nodes, including one start node, plural judgment nodes and plural processing nodes. The start node is to determine the first node to be executed. The judgment nodes are used to judge the information from environments and have conditional branch decision functions. Judgment nodes return the judgment result and decide the node to be executed next. Also the judgment in the judgment nodes is decided not only by the current situations but also the past situations. The processing nodes are used to determine the actions of GNP. GNP has network structures and each node has directed links to others. The basic structure of GNP is shown in **Figure 1**.

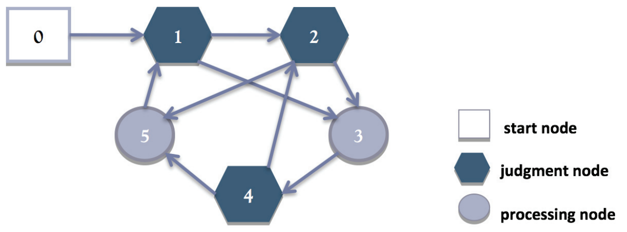

Fig. 1. The basic structure of Switch-GNP.

III. FUNCTIONALLY DISTRIBUTED GNP

The functionally distributed GNP [3] was used to compare with the proposed method, which consist of several GNPs doing their own simple task. Individuals of functionally distributed GNP include a Switch-GNP and several Sub-GNPs. The Switch-GNP controls Sub-GNPs, where each Sub-GNP corresponds to its own simple task. **Figure 2** shows the basic structure of the functionally distributed GNP.

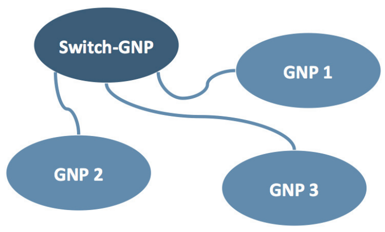

Fig. 2. The basic structure of Switch-GNP.

IV. PARALLEL GNP

The main feature of parallel GNP is that each GNP focuses on its own task, which makes the function of each GNP more specific to the task.

A. *Outline of Parallel GNP*

The proposed method named Parallel GNP is based on the increasingly booming parallel computing theory and applications. The parallel computing emphasizes that the large-scale task is decomposed into different sub-tasks, each task has its own work carried on a processor, and communications among processors should be considered if the works of several sub-tasks are not independent. We use the concept of parallel computing to design a new structure of GNP, i.e., parallel GNP. Several GNPs in the parallel GNP, which have their own task, work concurrently in order to realize the work consisted of several tasks. It means that several GNPs work like functionally distributed systems and when they conflict each other, one of the decisions produced by GNPs is chosen probabilistically. Different from the functionally distributed GNP, individuals of parallel GNP include several Sub-GNPs having their own task, where each Sub-GNP uses the basic GNP structure.

B. *Operation of Parallel GNP*

Suppose that a large and complicated task could be divided into two different tasks, i.e., $TASK_{Trash}$ and $TASK_{Energy}$ in one example. Each task has its own GNP, such as GNP_{Trash} or GNP_{Energy}. GNP_{Trash} and GNP_{Energy} can execute judgment or processing concurrently. There are three kinds of situations in one time step. There are three kinds of situations in one time step. If GNP_{Trash} executes judgment and GNP_{Energy} executes processing, the agent should take GNP_{Energy}'s action at this time step. On the contrary, if GNP_{Trash} executes processing and GNP_{Energy} executes judgment, the agent should take GNP_{Trash}'s action at this step. However, there occurs a conflict if GNP_{Trash} and GNP_{Energy} execute their own processing at the same time step. Then, the agent will choose one of them as a processing for the task. **Figure 3** and **Figure 4** shows the structure of Parallel GNP on how the GNP_{Trash} and GNP_{Energy} execute at the same time step.

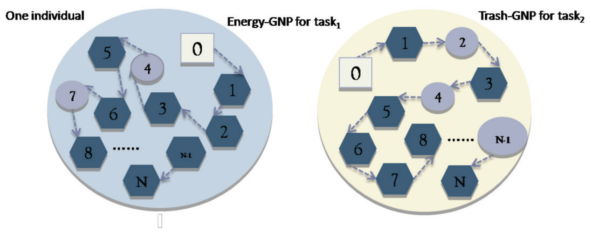

Fig. 3. Structure of Parallel GNP.

To deal with the conflict when parallel GNP executes, we use the probability Method to determine which processing should be taken. processing should be taken. Each GNP will be assigned probabilities of P_{Trash} and P_{Energy} (P_{Trash} + P_{Energy} =1). For example, we suppose $TASK_{Trash}$ is more important than $TASK_{Energy}$ according to the aim of the given task. Then, we generate a random value between 0.0 to 1.0. If the random value is smaller than P_s the processing of $TASK_{Energy}$ will be taken, on the other hand, if the random value is equal or larger than P_s, the processing of $TASK_{Trash}$ will be done by the agent, where P_s is a threshold probability less than 0.5.

The **Figure 5** shows the flow chart of Parallel GNP, which is explained as the following.

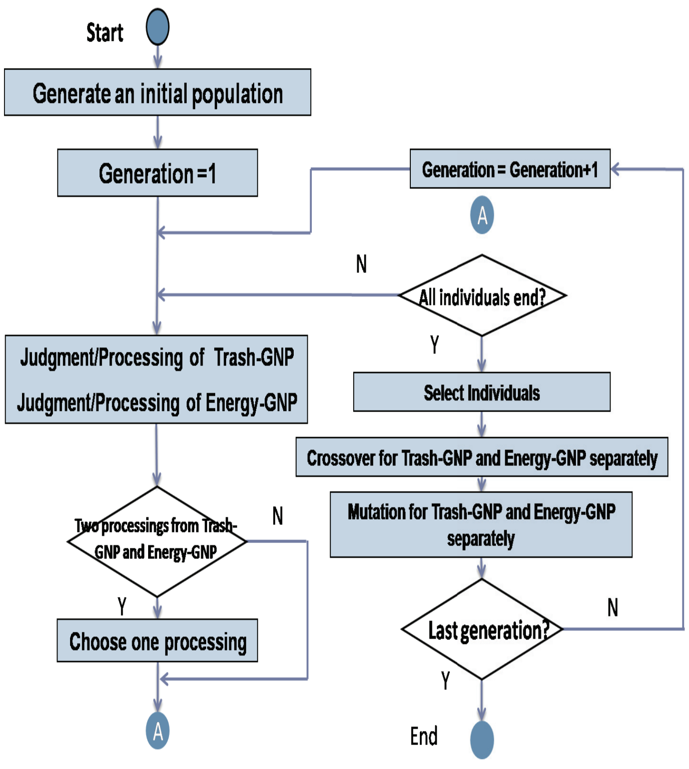
Start
Generate an initial population
Generation =1
Generation = Generation+1
A
N
All individuals end?
Y
Judgment/Processing of Trash-GNP
Judgment/Processing of Energy-GNP
Select Individuals
Crossover for Trash-GNP and Energy-GNP separately
Mutation for Trash-GNP and Energy-GNP separately
Two processings from Trash-GNP and Energy-GNP
N
Y
Choose one processing
Last generation?
N
Y
A
End

Fig. 5. Flow Chart of Parallel GNP.

$Step1 : Initialization$
N individuals with two GNPs are generated randomly.
$Step2 : Execution$
Two GNPs are executed at the same time, deciding which processing should be taken when the conflict occurs and each of the individuals in the current generation is evaluated.
$Step3 : EpisodeEndingCondition$
If the episodes of all individuals end, then goto genetic operations, otherwise, return to Step 2.
$Step4 : GeneticOperators$
Selection, crossover and mutation should be done.
$Step5 : TerminalConditions$
If the condition of the last generation is satisfied, the procedure of Parallel GNP will end, otherwise, return to Step 2.

C. *Evolution of Parallel GNP*

The style of evolution is not co-evolution, but conventional evolution, because the fitness function is defined for the system having several tasks, that is, the combination of the genes of GNP_{Trash} and GNP_{Energy} is regarded as the gene of parallel GNP .

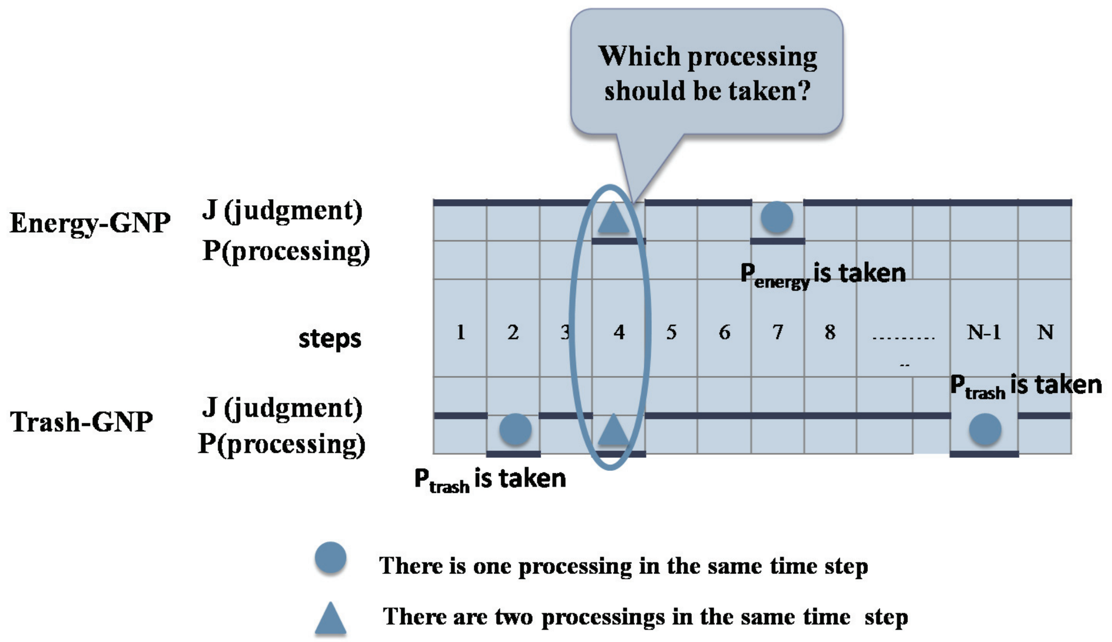
Which processing should be taken?
Energy-GNP
J (judgment)
P(processing)
steps
1 2 3 4 5 6 7 8 .... N-1 N
Trash-GNP
J (judgment)
P(processing)
P_{energy} is taken
P_{trash} is taken
P_{trash} is taken
There is one processing in the same time step
There are two processings in the same time step

Fig. 4. Execution of Parallel GNP.

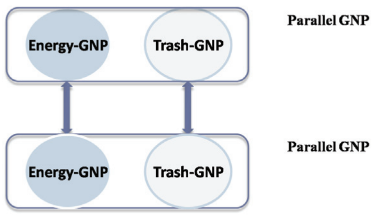

Fig. 6. Evolution of Parallel GNP.

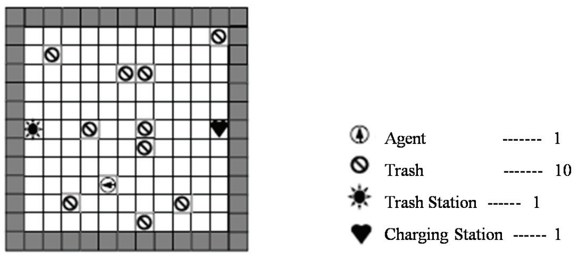

Fig. 7. Self-Sufficient garbage collector problem.

Parallel GNP includes selection, crossover and mutation as well. Especially, crossover is done between the same Sub-GNPs. **Figure 6** shows the evolution of the Parallel GNP.

V. SIMULATION

A. Simulation environment and parameters conditions

Parallel GNP has been applied to the self-sufficient garbage collector problem, where the objectives for agents are to collect more trash and move them to the collecting trash station and charge energy timely in the two dimensional grid made up of 13 by 13 cells(**Figure 7**). The self-sufficient garbage collector problem consists of one agent, ten trashes, a collecting trash station, and a charging station. The agent picks up trash when it gets to the cell with trash. There are four movements of agents, i.e., move forward, turn left, turn right and stay. Each action taken by the agent will lose energy. The energy consumption is like the agent loses three units of energy when it moves forward, one unit of energy when it turns left or right. The initial level of energy is 40. The agent charges energy by reaching the charging station and continuing to stay for a while. The amount of charging energy $\Delta E(t)$ is defined by Eq. (1).

$$\Delta E(t) = \frac{(E_{max} - E(t))^2}{2E_{max}}, \tag{1}$$

where,
$\Delta E(t)$: the energy level of the agent at time step t
E_{max}: the maximum energy level.

The fitness function F evaluates each individual from two aspects. One is the number of collected trash, and another is the current energy. The fitness for evolving parallel GNP is defined by Eq. (2).

$$F = \sum_{e \epsilon E} [T_e + \alpha \sum_{t \epsilon T} E_e(t)], \tag{2}$$

where,
T_e : the number of trash collected in environment e
$E_e(t)$: the energy level at time step t in environment e
E : set of suffixes of environments for training and testing
T : set of time steps
α : coefficient

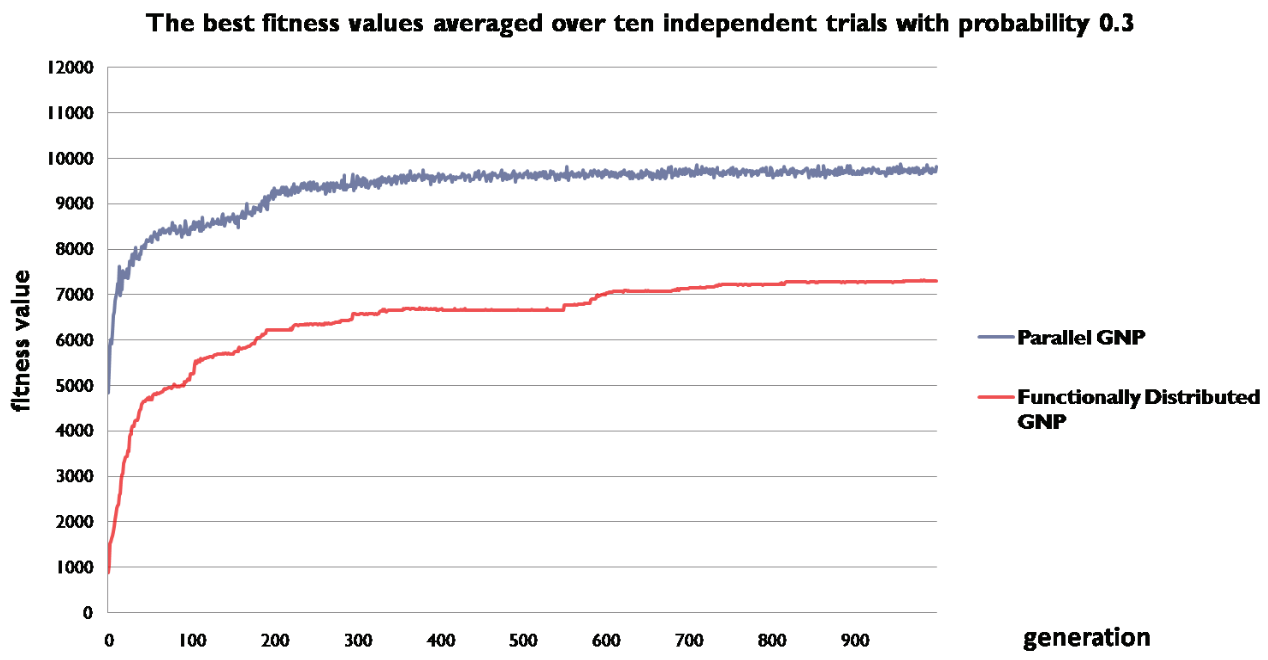

Fig. 8. The best fitness curves averaged over ten independent trials.

$E_e(t)$ of Eq. (2) of parallel GNP is for maintaining the energy level, because the agent cannot move if the energy is 0.

The judgment actions of $Trash-GNP$ are shown in **Table I**, while the judgment actions of $Energy - GNP$ are shown in **Table II**. The processing nodes of each GNP are the same, which are shown in **Table III**.

TABLE I
JUDGMENT FUNCTIONS OF TRASH-GNP

NODE TYPE	FUNCTION
1	Check how much trash the agent has taked
1	Find out the direction of the trash collection
1	Find out the direction of the nearest trash

TABLE II
JUDGMENT FUNCTIONS OF ENERGY-GNP

NODE TYPE	FUNCTION
1	Check the state of current energy
1	Find out the direction of the charging station

TABLE III
PROCESSING FUNCTIONS OF GNP

NODE TYPE	FUNCTION
0	Move forward
0	Turn left
0	Turn right
0	Stay

The parameters used in simulations are described in **Table IV**.

TABLE IV
PARAMETERS OF SIMULATIONS

ITEM	VALUE
Number of Generation	1000
Number of Individuals	100
Number of Elite Individuals	1
Crossover Size	40
Crossover Probability P_c	0.1
Mutation Size	59
Mutation Probability P_m	0.01

B. Simulation results

Figure 8 shows the average best fitness curves over ten independent trials of the Self-sufficient garbage collector problem in the training. It is found from simulation results that the best fitness value of the population is improved as the generation goes on. It is also clarified from **Figure 8** that Parallel GNP increases the fitness values faster than the conventional functionally distributed GNP and has the better solution in the end. But, there are some fluctuations in the fitness values of Parallel GNP, which are caused by the Probability Method. In the testing, the mean and standard deviation of the fitness values are calculated using 1000 different environments generated randomly using ten best evolved GNPs. In **Table V**, MAX show the maximum value of the fitness using 1000 different environments generated randomly. Therefore, **Table V** shows the generalization ability of the evolved GNPs. It is found from **Table V** that the more generalization ability is obtained by Parallel GNP rather than Functionally Distributed GNP.

TABLE V
GENERALIZATION ABILITY OF EVOLVED GNPS (FITNESS USING 1000 DIFFERENT ENVIRONMENTS)

METHOD	FITNESS
Functionally Distributed GNP	790.9
Parallel GNP	969.8

VI. CONCLUSION

In this paper, a new evolutionary computational method, that is, parallel GNP has been proposed for large and complicated systems. The proposed method introduces GNPs with their own task working concurrently in order to realize the work consisted of several tasks. It has been clarified from simulations of the Self-Sufficient garbage collection problem that the performances of the proposed method are improved in terms of collecting more trash and having higher fitness values.

REFERENCES

[1] R. A. Brooks, ”Robust layered control system for a mobile robot”, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, pp. 14-23, 1986.

[2] S. Mabu, K. Hirasewa and J. Hu, ”A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning”, Evolutionary Computation, Vol. 15, No.3, pp. 369-398, MIT Press 2007.

[3] S. Eto, H. Hatakeyama, S. Mabu, K. Hirasawa and J. Hu, ”Realizing Functional Localization Using Genetic Network Programming with Importance Index,” Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.10, No.4, pp. 555-566, 2006.