Humanoid Robot's Motion Planning using Genetic Network Programming

Y.X. Sun and H. Ogai

Ogai Research Lab., Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan
(Tel : +81-090-9581-4495)
(sand_sun@ruri.waseda.jp)

Abstract: We have developed the program for humanoid robot's whole body motion planning using Genetic Network Programming (GNP). First we introduce the main idea and the way constructing GNP, then the method of applying GNP to robot's whole body motion planning, and give the result of our research as a conclusion.

Keywords: Genetic Network Programming, population, individual, node, evaluation function

I. INTRODUCTION

Research on humanoid robot's motion planning has been drawing attentions since long ago. So far, there have already been many different methods solving this problem. These methods are mainly divided into three classes: motion planning based on motion capture technology; one using pin/drag interface based on Graphical User Interface (GUI); and motion planning in multi-dimensional space by specifying the initial posture and the final posture, then the system generates continues motions connecting these two, a typical method is rapid-exploring random trees (RRT) [1].

In the meantime, professor K.Hirasawa proposed an evolutionary algorithm: Genetic Network Programming (GNP) in the year of 2000 [2] and it's been gradually applied in various research fields to solve different kinds of practical problems like double-deck elevator group supervisory control systems, data mining, stock trading rules' generation and comprehensible control rules for real robots [3].

Inspired from these achievements, we tried effort to develop a control system for humanoid robots' motion planning using GNP. Basically, the goal is to generate control rules that manipulate robot's movements connecting a preset initial posture and a final posture, which in other words, is to find a new solution in the 3rd class of motion planning.

This paper is going to present the latest research progress. It is organized as follows: firstly, give a brief introduction to GNP in Section 2, then in Section 3, bring out the research model built based on MATLAB and detailed methods applying GNP to motion planning. Finally, give the result of our research and relative analysis as a conclusion.

II. GENETIC NETWORK PROGRAMMING

Nowadays, control systems are becoming large and complex that it is not easy for us human beings to figure out a clear and precise control rule in advance, so an idea of developing an intelligent system that could find optimal rules automatically popped out, and then, inspired from GP and GA, GNP was proposed.

1. Structure of GNP

The fundamental elements that construct a network for GNP are called nodes. Basically, two classes of nodes are required: processing node and judgment node.

Processing nodes describe the actions of GNP and judgment nodes judge the information from the environment and choose to take actions according to the judged result. They connect with each others to form a control network. This kind of network is called an individual. Fig.1 shows a typical structure of one individual of GNP.

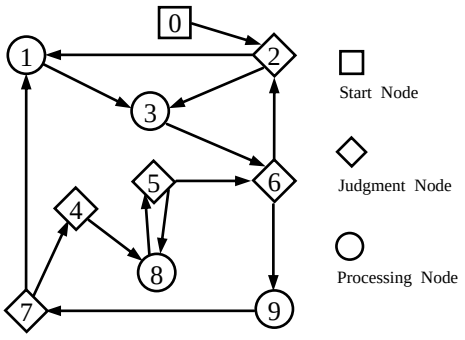

Fig.1 Basic structure of GNP

Fig.1 illustrates the phenotype of GNP's network structure, while Fig.2 shows the genotype of it:

Node 0	0	0	0	2	0	
Node 1	2	1	5	3	0	
Node 2	1	1	1	3	0	1 0
Node 3	2	2	5	6	0	
Node 4	1	1	1	2	0	8 0
Node 5	1	3	1	8	0	6 0
Node 6	1	2	1	9	0	2 0
Node 7	1	2	1	1	0	4 0
Node 8	2	1	5	5	0	
Node 9	2	2	5	7	0	

NT_i	FID_i	d_i	C_{i1}	d_{i2}	……	C_{in}	d_{in}

Fig.2 Genotype of Fig.1

We use a two-dimensional array to establish the gene structure of an individual. Each row represents one node, could either be a processing node or a judgment node, it depends on the value of the first element in the row array: NT_i. In the above table, for example, NT_i equals 1 meaning a judgment node while 2 meaning a processing node.

FID_i describes the function of the i^{th} node and d_i presents the time cost (delay) for the node to carry out its action or judgment.

After deciding the node's type, function and delay time, we need to decide which node does the present node connect. Here C_{in} denotes the connections and d_{in} denotes the connection time delay. Note that if the node is a processing node, it can only have one connection and if it is a judgment node, the number of connections can be more than one.

2. Genetic Operations in GNP

There are three types of the genetic operations in GNP: **selection**, **crossover** and **mutation**.

Selection: Individuals are initialized randomly at the beginning of the program. After mutation or crossover, apparently the population becomes larger. The selection operation can help the program select the individuals with better performance among the population and keep it within a fit size.

We take two ways to carry out selection operation: tournament selection and elite selection. Tournament selection is carried out between any two individuals. The program calculates two individuals' fitness and dumps the low valued one. Elite selection calculates all individuals' fitness and sets a bottom line of it. Ones that are higher than the line are chosen and the rest are discarded.

Crossover exchanges the sub-network of two parent individuals to create new individuals. Here the sub-network can be several nodes which connect each other, can be only one point, or just several points which are randomly chosen from parents.

Mutation takes effect in a single individual. It is used to change one node's connection branches, the function, or even its node type. This is another way of producing new kind of individuals.

Both crossover and mutation offers new individuals in a population for selection to select better ones.

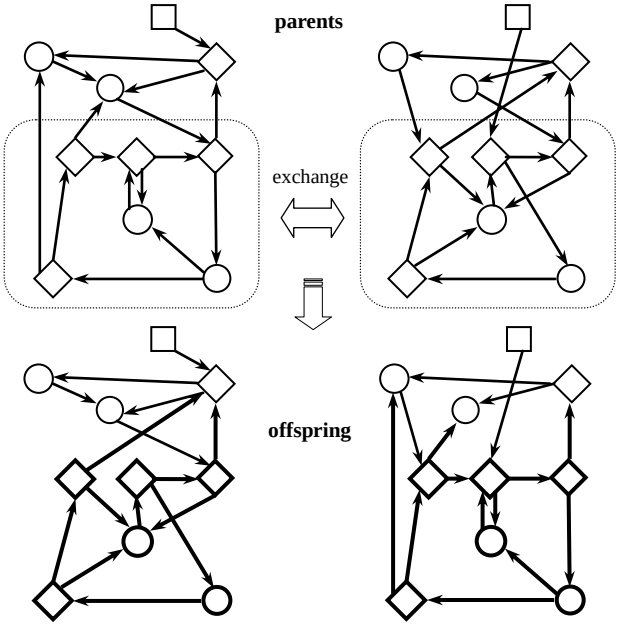

Fig.3 Crossover in GNP

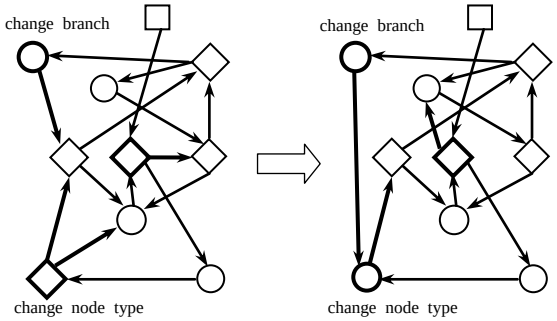

Fig.4 Mutation in GNP

III. MODEL AND APPLICATION DESIGN

1. Modeling

We built a model of humanoid robot which is illustrated in Fig.5. It has 19 degrees of freedom and

each joint's range of rotatable angle has been specified generally according to human beings.

The data structure of the robot is a tree structure, each joint has a sister branch and a child branch. Start from the head, we can use forward kinematics to calculate the posture of the model robot. Besides, with the mass of each joint set during the initialization, calculation of the robot's center of mass is possible.

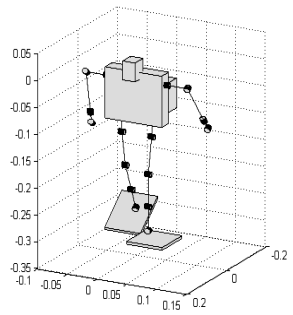

Fig.5 Model of a humanoid robot

2. Application Design

In this chapter, we are to develop an application that can find optimal control rules to generate motion series which connecting the preset initial posture and final posture. We start from designing the various nodes (type and related functions) that are intended for constructing individuals.

A. Processing Node

Since the model we built has 19 degrees of freedom, the actions that processing nodes take are to manipulate the joint angles of these 19 joints. Each node takes charge of one joint and can only choose to adjust its angle to plus or minus a certain value. So if we want the entire body of the robot be controlled, at least 38 kinds of processing nodes are needed (2 nodes, respectively, doing plus and minus for each joint).

For more adaptive control, even the step length of the adjustments can have many choices, like one degree per action or two. But by doing this, the kinds of processing node grow extremely large and system becomes complex, so here in our research, we just take the simplest way: 38 kinds of processing nodes.

In order to make sure that the robot can finally reach the desired posture, we set a constant command in all processing nodes, that is: after taking its action, the joint is forced to get closer by one degree to its final value.

B. Judgment Node

During the robot's movement, the most important thing is to keep balanced. Here we suppose our robot's movement is slow and steady, so the momentum which might affect the robot's balance is not considered. Then, the only element relative to balance is the center of mass (CoM). So we design the judgment conditions to be: which quadrant does the CoM locate in X-Y coordinate system, and then choose actions according to the judged result.

Table.1 Node Assignment

Node Type	**Functions**		
Processing Node	Joint 1 + 1°	Joint 1 - 1°	
	Joint 2 + 1°	Joint 2 - 1°	
	… …	… …	
	Joint 19 + 1°	Joint 19 - 1°	
Judgment Node	**Node Kind**	**X-Axis**	**Y-Axis**
	1	+	+
	2	+	-
	3	-	+
	4	-	-

C. Evaluation Function

The evaluation function is used to calculate the fitness of each individual. There are two parts that add up to present the fitness value: one is the total steps that the robot's all 19 joints took to perform the desired motion; the other is the CoM's average deviation from the track connecting the CoM at start and the one at the end, which is easy to understand by referring to Fig.6.

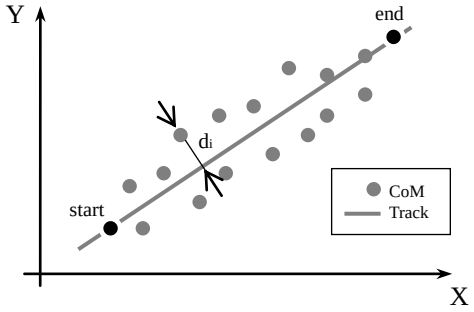

Fig.6 Deviation of CoM from desired track

Evaluation Function:

$$\text{fitness} = n * \alpha + \frac{\sum d_i}{n} * \beta$$

where n denotes the step count, $\sum d_i / n$ calculates the average deviation, and α, β are weight coefficients.

One point should be clarified is that, the desired track of the CoM is not necessarily to be a straight line, a curve might do better in some circumstances. Here in our experiment, we take the track as a straight line.

Once all the three elements are decided, the genetic operations can be carried out. Fig.8 illustrates the flow chart of the application.

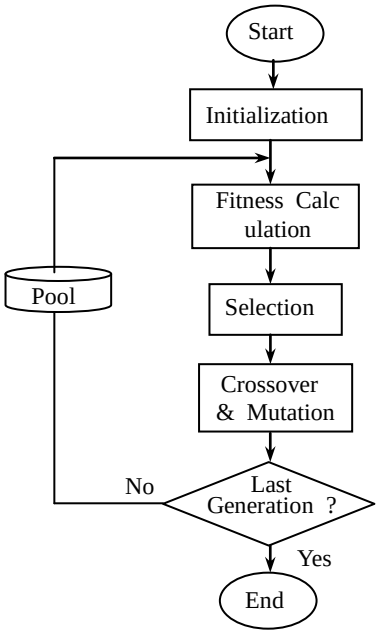

Fig.7 Flow Chart of Program

In case that too many individuals were discarded after selection, we here set a data pool storing randomly generated individuals to relief the population crisis that might happen in chance.

IV. EXPERIMENT RESULT

Considering that the number of node functions is large, during the initialization period, if we directly let the computer generate all the individuals randomly, maybe none of them has the ability to simply control all the joints. Though after genetic operations, there might be several ideal individuals produced, we decided to help accelerate the evolution procedure by adding several man-made individuals that has the ability to control all the joints. Then let the genetic operations be carried out. Table.2 shows the parameters of the program:

Table.2 Parameters Specifications

Selection Rate	0.24
Crossover Rate	0.08
Mutation Rate	0.08
Population(Individuals)	25
Judgment Node	20
Processing Node	80
Starting Node	1
Generation	30

After trained the population using the above parameters by several target motions, we obtained some individuals that can control the robot to perform some simple movements. Then we went on to train these individuals for a test motion, we found that there is still space for improvement referring to Fig.8 which shows the fitness value curve as generations grow.

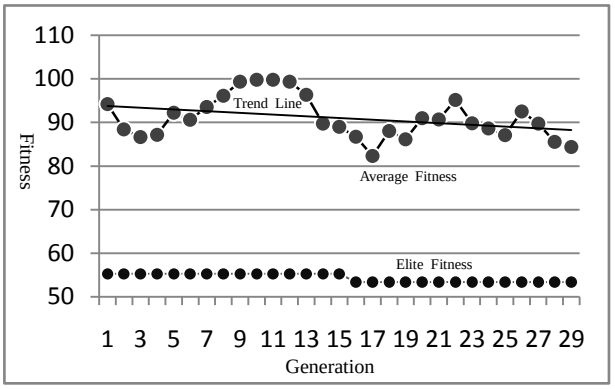

Fig.8 Fitness Curve

Using the elite individual selected from the last generation, we control the model robot to perform a simple motion of "kick" whose snapshots are shown in Fig.9.

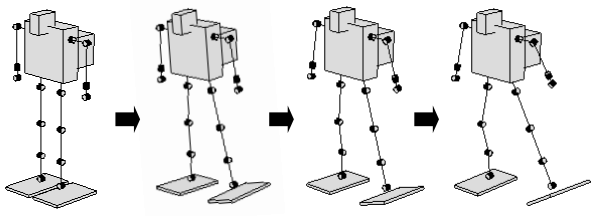

Fig.9 Snapshots of "kick"

V. CONCLUSION

So far, we have developed an application using GNP algorithm to generate control rules for a humanoid robot. But there is still space for improvements: to generate complex motions by setting several transitional key postures; to add new kinds of judgment node judging momentum and as mentioned in Section.4: in the initialization phase, we still have to help computer generate the first generation, or it will take too much time to finally obtain a usable individual, so how to improve the efficiency during initialization phase will also be our future research target.

REFERENCES

[1] Kajita S (2005), Humanoid Robot (in Japanese). pp.163-170

[2] Katagiri H, Hirasawa K, Hu J (2000), Genetic Network Programming – Application to Intelligent Agents. IEEE pp.3829-3834

[3] Murata T and Okada D (2006), Using Genetic Network Programming to Get Comprehensible Control Rules for Real Robots. 2006 IEEE Congress on Evolutionary Computation, pp.1983-1988