Elevator Group Supervisory Control System Using Genetic Network Programming with Macro Nodes and Reinforcement Learning

Jin Zhou Non-member  (Waseda University, zhoujin@asagi.waseda.jp)
Lu Yu Non-member  (Waseda University, yulu1006@fuji.waseda.jp)
Shingo Mabu Member  (Waseda University, mabu@waseda.jp)
Kotaro Hirasawa Member  (Waseda University, hirasawa@waseda.jp)
Jinglu Hu Member  (Waseda University, jinglu@waseda.jp)
Sandor Markon Non-member  (Fujitec Co. Ltd., ms@fujitec.com)

Keywords: elevator group supervisory control system, genetic network programming, reinforcement learning, importance weight

GA evolves strings and it is mainly applied to optimization problems, and GP was devised later in order to expand the expression ability of GA by using tree structures. This structural change of solutions brought progress on the evolutionary computation and made GP applicable to more complex problems. Since the past studies suggested that different structure has different expression ability, a new evolutionary computation method called Genetic Network Programming (GNP) with directed network structures has been proposed. To verify its applicability and efficiency, some studies have been done on both virtual and real world problems.

GNP was firstly applied to Elevator Group Supervisory Control System (EGSCS) in a real world problem after its applicability and efficiency has been clarified in virtual world such as tile-world model and ant colonies, etc. Simulation test results demonstrated that GNP could also work well on such a complex stochastic optimal control problem. However, even though some improvements of the EGSCS' performances over the conventional control methods have been made using GNP, there remain some problems such as searching for faster training speed and better performances. In EGCS using only GNP, the GNP structure and its parameters are optimized by genetic operations at the end of each generation. That is to say, there is nothing optimized during the individual execution process. On the other hand, an extended algorithm of GNP combining learning and evolution called Genetic Network Programming with Reinforcement Learning (GNP with RL) has been proposed and its efficiency has been also verified with some benchmark problems. Comparing to the benchmark problems such as tile-world, EGSCS is a more complex real world problem. In this paper, we should study an appropriate algorithm based on the inherent nature of EGSCS because the method using GNP with RL in benchmark problems can not be employed in EGSCS directly and easily. Therefore, in this paper, we propose a new method for EGSCS using GNP with Macro Nodes and Reinforcement Learning (GNP-RL) as shown in Fig. 1.

With only the GNP system, evolution takes place after a set of simulation runs are completed and the fitness of each individual GNP is evaluated. In contrast with this, when RL is added, the system could perform online learning even during task execution. Thus, we could expect faster learning and improved performances. The performance of the proposed method is studied by simulations under several conditions. Some analyses are made based on these test results comparing to other algorithms using original GNP and conventional control methods. Moreover, to explore further the efficiency of the proposed method, we optimize the framework of GNP with RL by tuning the importance weight of the macro-processing node. And the experiment results show that some better performances are obtained with the importance weight optimization employed.
Elevator Group Supervisory Control System Using Genetic Network Programming with Macro Nodes and Reinforcement Learning

Jin Zhou* Non-member
Lu Yu* Non-member
Shingo Mabu* Member
Kotaro Hirasawa* Member
Jinglu Hu* Member
Sandor Markon** Non-member

Elevator Group Supervisory Control System (EGSCS) is a very large scale stochastic dynamic optimization problem. Due to its vast state space, significant uncertainty and numerous resource constraints such as finite car capacities and registered hall/car calls, it is hard to manage EGSCS using conventional control methods. Recently, many solutions for EGSCS using Artificial Intelligence (AI) technologies have been reported. Genetic Network Programming (GNP), which is proposed as a new evolutionary computation method several years ago, is also proved to be efficient when applied to EGSCS problem. In this paper, we propose an extended algorithm for EGSCS by introducing Reinforcement Learning (RL) into GNP framework, and an improvement of the EGSCS’ performances is expected since the efficiency of GNP with RL has been clarified in some other studies like tile-world problem. Simulation tests using traffic flows in a typical office building have been made, and the results show an actual improvement of the EGSCS’ performances comparing to the algorithms using original GNP and conventional control methods. Furthermore, as a further study, an importance weight optimization algorithm is employed based on GNP with RL and its efficiency is also verified with the better performances.

Keywords: elevator group supervisory control system, genetic network programming, reinforcement learning, importance weight

1. Introduction

As typical evolutionary computation methods, Genetic Algorithms (GA) (1), Genetic Programming (GP) (2) (3) and Evolutionary Programming (EP) (4) (5) have been proposed, and a large number of studies applying them to many fields have been conducted. As we know, GA evolves strings and it is mainly applied to optimization problems, and GP was devised later in order to expand the expression ability of GA by using tree structures. This structural change of solutions brought progress on the evolutionary computation and made GP applicable to more complex problems. Since the past studies suggested that different structure has different expression ability, a new evolutionary computation method called Genetic Network Programming (GNP) (6) (7) with directed network structures has been proposed. To verify its applicability and efficiency, some studies have been done on both virtual and real world problems.

GNP was firstly applied to Elevator Group Supervisory Control System (EGSCS) in a real world problem (8) after its applicability and efficiency has been clarified in virtual world such as tile-world model and ant colonies, etc. Simulation test results demonstrated that GNP could also work well on such a complex stochastic optimal control problem. However, even though some improvements of the EGSCS’ performances over the conventional control methods have been made using GNP, there remain some problems such as searching for faster training speed and better performances. In EGSCS using only GNP, the GNP structure and its parameters are optimized by genetic operations at the end of each generation. That is to say, there is nothing optimized during the individual execution process. On the other hand, an extended algorithm of GNP combining learning and evolution called Genetic Network Programming with Reinforcement Learning (GNP with RL) (7) (9) has been proposed and its efficiency has been also verified with some benchmark problems. Comparing to the benchmark problems such as tile-world, EGSCS is a more complex real world problem. In this paper, we should study an appropriate algorithm based on the inherent nature of EGSCS because the method using GNP with RL in benchmark problems can not be employed in EGSCS directly and easily. Therefore, in this paper,
we propose a new method for EGSCS using GNP with Macro Nodes and Reinforcement Learning (GNP-RL). With only the GNP system, evolution takes place after a set of simulation runs are completed and the fitness of each individual GNP is evaluated. In contrast with this, when RL is added, the system could perform synchronous learning even during task execution. Thus, we could expect faster learning and improved performances. The performance of the proposed method is studied by simulations under several conditions. Some analyses are made based on these test results comparing to other algorithms using original GNP and conventional control methods. Moreover, since different evaluation items have different “importance” when making car assignment based on the current state of EGSCS, fixing the importance weight of each evaluation item on an identical value is not enough. Thus, we optimize the framework of GNP with RL by also tuning the importance weight of the macro-processing node to explore further the efficiency of the proposed method. And the experiment results show that some better performances are obtained with the importance weight optimization employed.

This paper is organized as follows. The next section describes the outline of GNP with RL. In section 3, the proposed method is explained. Section 4 shows the simulation conditions and results. Two experiments are made to verify the efficiency of GNP with RL and the importance weight optimization algorithm, respectively. Finally, some conclusions and future work are devoted in section 5.

2. Genetic Network Programming with Reinforcement Learning

In this section, we take a brief overview of GNP with RL. The original GNP is based on the general evolutionary framework such as selection, crossover and mutation. GNP with RL is based on evolution and reinforcement learning. The aim of combining evolution and RL is to make good programs using the current information (state and reward) during task execution in dynamic environments. Since evolution based algorithms use only the fitness values calculated after finishing a task, GNP with RL has an advantage that much of the required information can be utilized during task execution.

2.1 Structure of GNP with RL  Figure 1 shows the basic structure of GNP with RL. Like the original GNP, GNP with RL also consists of three basic elements, i.e., nodes, branches and time delays. All features of original GNP such as directed graph expressions, reusability of nodes and storage of neutral genes are inherited in GNP with RL. A main difference from the original GNP is that the judgment/processing node is extended to a macro one to implement the learning process. Thus, each node of the original GNP has only one function, while in GNP with RL each node could have several functions and one of them is selected based on the policy.

In Fig.1, $K_i$ represents the node type, which is the same as original GNP. $W_i$ is proposed in this paper as the importance weight of each node and it can be optimized through the mutation process described later. $D_{ip}$ means $Q$ value which is assigned to each state and action pair. In this method, “State” means a current node, and “Action” means a selection of node function $D_{ip}$. Note that the actions of agents and the actions in the RL of GNP are not the same. $d_{ij}$ is the time delay spent on judgment or processing. $C_{ip}^A, C_{ip}^B, \ldots$ show the node number of the next node $j$. $d_{ip}^A, d_{ip}^B, \ldots$ mean time delays spent on the transition from node $i$ to node $j$.

2.2 Running Process of GNP with RL  As mentioned above, GNP with RL combines the evolution and learning processes to take advantage of the sophisticated diversified search ability of evolution, and the intensified search and synchronous learning abilities of reinforcement learning. Fig.2 shows the outline of evolutionary and learning processes of GNP with RL. Reinforcement learning process is executed during task execution of each individual and the learning results are encoded in GNP genes which are inherited to the next generation after the genetic operators are executed.

2.2.1 Evolutionary Process  GNP with RL also has three kinds of genetic operators, i.e., selection, crossover and mutation. All of them except mutation are the same as original GNP.

• Selection

The commonly used selection operators in evolutionary computations are “Roulette Selection”, “Ranking Selection”, “Tournament Selection” and “Elite Preservation Selection”. In this paper, we use the latter two in GNP with RL.

• Crossover

We use the “Uniform Crossover”, which is executed between two parents and generating two offspring. All gene information of the two corresponding nodes with the same node number is exchanged between two parents including the reinforcement learning results. The crossover procedure is as follows:

(1) Select two individuals as parents using tournament selection twice.

(2) Each node is selected as a crossover node with the probability of $P_c$.

(3) Two parents exchange the genes of the corresponding crossover nodes with the same node number.

(4) The generated new individuals become the new ones of the next generation.

• Mutation

In GNP with RL, mutation operation could be executed not only on the connections among nodes but

† When not adding importance weights to each node which presents one of the evaluation items, all evaluation items are considered uniformly when assigning an emerged hall call to a car. However, in fact, the importance of these items might be different.
also on the type and number of a macro-node since the macro-node has more than one sub-nodes with different type of node functions. For simplicity, in this paper, we just only execute the mutation operation on connections. In order to optimize the importance weight of each macro-processing node, step (2a) will be executed during the mutation operation, while skipped when the importance weight is set at a constant.

(1) Select one individual as a parent using tournament selection.

(2) Change the connections of each node with the probability of \( P_m \).

(2a) Tune the importance weight of each macro-processing node. Here, we update the importance weight by ± 0.1 with probability 0.5.

(3) The generated new individual becomes the new one of the next generation.

2.2.2 Learning Process As mentioned above, a state means a current node and an action means the selection of a function. Fig. 3 shows states, actions and an example of node transition. Learning process is explained as follows based on that example.

(1) At time \( t \), GNP refers to \( Q_{i1}, Q_{i2}, ..., Q_{im} \), and selects one of them based on \( \varepsilon \)-greedy policy. We suppose that GNP selects \( Q_{ip} \) and the corresponding function \( ID_{ip} \).

(2) Then GNP executes the function \( ID_{ip} \), gets the reward \( r_i \) and the next node \( j \) becomes \( C_{ip}^A \).

(3) At time \( t+1 \), GNP selects one \( Q_{jp} \) in the same way as step 1.

(4) Then the following procedure is executed.
\[
\delta = r_i + \gamma Q_{jp} - Q_{ip} \\
Q_{ip} \leftarrow Q_{ip} + \alpha \delta
\]

(5) \( t \leftarrow t + 1, i \leftarrow j, p \leftarrow p' \) then return step 2.

In this example, node \( i \) is a macro-processing node, but if it is a macro-judgment node, the next current node is selected among \( C_{ip}^A, C_{ip}^B, ... \) according to the judgment result.

3. Application of GNP with RL to EGSCS

3.1 Elevator Group Supervisory Control System (EGSCS) Elevator Group Supervisory Control System (EGSCS) is a very large-scale stochastic optimization problem. The task of an elevator group system is to carry the passengers efficiently in a vertical
transportation system. In general, an EGSCS consists of a high-rise building with several elevators installed, a dynamic passenger flow and a group control system. Unlike other usual transportation systems like train systems which run on a pre-scheduled time table, the EGSCS is driven by the passenger flow with some actions such as pushing hall/car buttons and getting on/off elevator cars. In an elevator group system, there are some events emerging probabilistically like hall calls (passenger arrival) and car calls (passenger destination). Moreover, the EGSCS is also a partially observable dynamic system since the number of passengers waiting at a floor is unknown as well as the number of passengers inside an elevator car who are getting off at a floor. There are two kinds of car assignment policies. One is called immediate policy, which makes the car assignment immediately after a new hall call occurs. Although this policy is sub-optimal, but it is helpful for palliating the tension of waiting passengers by making the car assignment immediately and giving a guidance message to passengers. The other is called reassignment policy, which makes the car assignment in an appropriate timing or reassigns the past car assignment. It tends to find the optimal solution using the latest information of the elevator group system, but much more computational costs are needed.

Different buildings have different passenger traffic patterns. In a typical office building usually studied as an objective, there are three types, i.e., “Up Peak Time”, “Down Peak Time” and “Regular Time”. The usual performance criterions of an EGSCS include the average waiting time (AWT) of all passengers in the system, the long waiting ratio (LWR) and maximum waiting time (MWT), and so on. For those reasons mentioned above, it is quite difficult to optimize one or more of these performance criterions by only employing classical control methods. Recently, Artificial Intelligence (AI) technologies are used in this field to make an “intelligent” controller. The algorithm using fuzzy-logic rules makes a significant improvement and is applied to some real elevator group systems. The good performance, however, seems to be tightly related to the expert rules making it hard to be employed widely. Another approach using reinforcement learning based on a neural networks (NNs) framework is reported in. Although the algorithm using Q-learning performed slightly better, i.e., by 2.65% than FIM (Finite Intervisit Minimization) and ESA (Empty the System Algorithm) for one specific down-peak pattern, it took 60,000 hours for the simulated elevator operation to converge, which is not practical for real elevator systems. Recently, a new approach using GNP was proposed in EGSCS and made a promising improvement of performance. In this paper, we propose an extended algorithm using GNP with RL.

3.2 EGSCS using GNP with RL Figure 4 shows the structure of the proposed method. In this system, a GNP individual from the genetic pool acts as the elevator group controller and makes a car assignment when a new hall call occurs. A best individual will be selected as the controller when the evolutionary and learning process end. Here, we use the immediate policy in our proposed controller. Moreover, we employ a compulsory dispatch strategy in up peak mode, where the idle cars will be compulsorily dispatched to the Terminal floor (or, main lobby).

3.2.1 Evaluation Items In EGSCS, many factors, such as predicted arriving time, the number of loaded passengers, the number of registered hall/car calls, and so on, should be considered when making an appropriate car assignment. As we know, it is important to know some apriori knowledge to determine evaluation items when designing the EGSCS controller. In this paper, we take the following six factors as the evaluation items, which are calculated based on the information of EGSCS every time the new hall call occurs.

- ATI – Estimated arriving time of a car after a new hall call assignment considering the incremental waiting time of other registered hall calls scheduled to be served by the same car. ATI directly relates to the passenger’s waiting time, the smaller, the better. That is to say, the car with the smallest value of ATI will firstly arrive at the floor where the new hall call occurs, which means those passengers can be served within a shortest waiting time if this car is assigned to the new hall call.
- PN – The number of passengers loaded in a car. It represents the current transport capacity of a car, the smaller, the better. That is to say, assigning the car...
with the smallest value of PN to the new hall call will tend to balance the transport load of all cars.

HCN – The number of registered hall calls scheduled to be served by a car. It represents the future transport capacity of a car, the smaller, the better since a large HCN means many more passengers will board this car.

CCN – The number of registered car calls scheduled to be served by a car. It also represents the future transport capacity of a car, the larger, the better since a large CCN means many more passengers will get off this car.

MC – Whether or not registered car calls coincide with a new hall call. Coincidence is better. That is to say, the new hall call could be served by the car which has such a coincidence without an additional stop, improving the EGSCS transport efficiency in a sense.

BM – Whether or not the fastest arriving car and the second fastest one are running in a bunching mode. If they are, it is better to assign the second fastest car to a new hall call. Since BM is linked to a special running mode, we deal with it as one independent evaluation item even though it has some potential relations with ATI and MC.

3.2.2 Node Functions In this paper, we proposed one kind of judgment node and two kinds of processing nodes except for the boot node. When a new hall call occurs, the GNP-RL network will be activated starting from the boot node and transfer to one of the judgment nodes \( J_1 \). All judgment nodes \( J_1 \) described later form the hall call judgment part whose branches connected to processing nodes \( P_1 \), which form the car candidate selection part. After the car candidates are selected in this part, a car will be determined to serve the new hall call in the car assignment part which consists of processing nodes \( P_2 \). The node transition will start again and transfer to one of the judgment nodes \( J_1 \) when the new hall call occurs. Functions of judgment and processing node are defined as follows.

- **Processing Node**

  \( P_1 \): A macro node with six sub-nodes in it. Each sub-node corresponds to an evaluation item described above, respectively. It will determine a car by the selected sub-node as a candidate to be assigned to the new hall call.

  \( P_2 \): Eq. (1) shows that processing node \( P_2 \) is to make the car assignment for a new hall call based on the candidate set created by node \( P_1 \) transition. The candidate cars selected by different node \( P_1 \) could have different importance weights, and processing node \( P_2 \) can make the car assignment by considering the importance weight of each node in \( P_1 \). The importance weight could be optimized during the evolutionary and learning process. In this paper, it is set at a constant in GNP-RL without weight tuning to check the efficiency of the proposed method of introducing RL into GNP, and then, in GNP-RL with weight tuning, we attempt to optimize it by tuning its value.

  \[ d = \arg \max_{l \in L} c(l) \]

  \[ c(l) = \sum_{j \in J} w_j s(l, j) \]

  \[ s(l, j) = \begin{cases} 1 & \text{if the output of node } j \text{ is car number } l \\ 0 & \text{otherwise} \end{cases} \]

  where, \( L \) : set of suffixes of cars, \( J \) : set of suffixes of passed macro-processing nodes during node transition of \( P_1 \), \( s(l, j) \): importance weight of macro-processing node \( j \), \( d \): a new hall call is assigned to car number \( d \).

- **Judgment Node**

  \( J_1 \): Judge which type a new hall call is from the information on the floor where a new hall call occurs. Since the new hall call has some implicit attributes corresponding to its location and direction, we add this kind of judgment node to identify the type of the new hall call expecting to search an optimal car assignment for each type of hall calls. There are five types of judgment results, i.e., Up direction call at terminal floor, Down direction call at low general floor, Up direction call at low general floor, Down direction call at high general floor, Up direction call at high general floor.

3.2.3 Reward Function We define the reward \( r \) of Q-learning process as the function of waiting time. Since one of the final goal in the proposed method is to minimize the average waiting time, a short waiting time by a car assignment action can be regarded as a reward (a large reward value), while a long waiting time is regarded as a punish (a small reward value). Then, the reward function is defined as follow.

  \[ r = e^{-kT}, \quad \text{where, } T \text{ is the waiting time of the new hall call, and } k \text{ is the coefficient of } T. \]

3.2.4 Fitness Function Comparing to reward function, the fitness function is employed to evaluate the performance of a GNP-RL individual in a whole during task execution, so it considers the average waiting time and the maximum waiting time instead of the waiting time of each hall call. Moreover, since there might be some “loops” formed in the GNP-RL network, the fitness function also should consider a term to evaluate the loop of the GNP-RL individual. The fitness of the proposed method is defined by Eq. (3). The former two terms in the right hand are used to minimize the average waiting time and the maximum waiting time. The third one is used to eliminate the genes causing “loop” of the node transition. The cases where the accumulated time delay becomes greater than the predefined threshold are also considered as loop.

  \[ f = \frac{1}{|P|} \sum_{p \in P} (t_p)^2 + w_t \times (t_{\text{max}})^2 + w_n \times (n)^2, \quad \text{where, } |P| \text{ is the total number of passengers during the simulation time, } t_p \text{ is the waiting time of passenger } p \in P, t_{\text{max}} \text{ is the maximum waiting time, } n \text{ is the number of the loops, and } w_t \text{ and } w_n \text{ are the coefficient of } t_{\text{max}} \text{ and } n, \text{ respectively.} \]
Table 1. Specifications of EGSCS Simulator

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Floors</td>
<td>16</td>
</tr>
<tr>
<td>Number of Elevators</td>
<td>6</td>
</tr>
<tr>
<td>Floor Distances</td>
<td>4.5 [m]</td>
</tr>
<tr>
<td>Max. Car Speed</td>
<td>2.5 [m/s]</td>
</tr>
<tr>
<td>Max. Car Acceleration</td>
<td>0.7 [m/s²]</td>
</tr>
<tr>
<td>Car Capacity</td>
<td>20 [persons/car]</td>
</tr>
<tr>
<td>Time Spent on</td>
<td></td>
</tr>
<tr>
<td>Opening Door</td>
<td>2.0 [s]</td>
</tr>
<tr>
<td>Closing Door</td>
<td>2.3 [s]</td>
</tr>
<tr>
<td>Riding/Leaving</td>
<td>1.0 [s/person]</td>
</tr>
<tr>
<td>Traffic Density</td>
<td></td>
</tr>
<tr>
<td>Regular, Down Peak</td>
<td>2000 [persons/hour]</td>
</tr>
<tr>
<td>Up Peak</td>
<td>1800 [persons/hour]</td>
</tr>
</tbody>
</table>

Table 2. Ratio of Passengers in Each Traffic Flow

<table>
<thead>
<tr>
<th></th>
<th>Regular</th>
<th>Up Peak</th>
<th>Down Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor (TF)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminal</td>
<td>5</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>General</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3. Parameters for GNP with RL

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation</td>
<td>300</td>
</tr>
<tr>
<td>Population Size</td>
<td>300</td>
</tr>
<tr>
<td>Mutation Size</td>
<td>170</td>
</tr>
<tr>
<td>Crossover Size</td>
<td>120</td>
</tr>
<tr>
<td>Elite Size</td>
<td>10</td>
</tr>
<tr>
<td>Node Size</td>
<td>31</td>
</tr>
<tr>
<td>Processing Node P1</td>
<td>10(10/Kind)</td>
</tr>
<tr>
<td>Processing Node P2</td>
<td>10(10/Kind)</td>
</tr>
<tr>
<td>Judgment Node J1</td>
<td>10(10/Kind)</td>
</tr>
<tr>
<td>Boot Node</td>
<td>1</td>
</tr>
<tr>
<td>Mutation Probability Pm</td>
<td>0.01</td>
</tr>
<tr>
<td>Crossover Probability Pc</td>
<td>0.1</td>
</tr>
<tr>
<td>Learning Coefficient α</td>
<td>0.1</td>
</tr>
<tr>
<td>Discount Ratio γ</td>
<td>0.7</td>
</tr>
<tr>
<td>ε-greedy ε</td>
<td>0.1</td>
</tr>
</tbody>
</table>

4. Simulation Results

4.1 Simulator Specifications Table 1 shows the specifications of an EGSCS simulator which is created as our test-bed. In this paper, we verify the efficiency of the proposed method with a typical office building, having 16 floors and 6 elevator cars installed. As shown in Table 2, three traffic patterns are used in the simulation test. The rows of the table represent the floor where passengers emerge, and the columns represent the floor where passengers plan to go.

4.2 Execution Conditions Parameters of the proposed method are listed in Table 3. Note that the size of generation and population is set at 500 and 310 respectively in some of our past studies. Since we found that there are almost no improvements of fitness after 300th generation, or the tendency to an overfit learning even if there are some improvements, we make the simulation run with 300 generations and 300 individuals in each generation for the same reason. The node size could be evolved for searching an appropriate one during the evolutionary process. In this paper, we proposed one kind of judgment node and two kinds of processing nodes, with 10 nodes of each kind. Moreover, the node size is also fixed for simplicity.

4.3 Results and Discussions In this section, we did experiments to study the proposed method from two viewpoints. First, to verify the efficiency of the proposed method using GNP with RL, we make the GNP-RL without weight tuning by fixing the importance weight at 1.0 for each macro-processing node. And after that, the GNP-RL with weight tuning is made where we optimize the importance weights of all macro-processing nodes during the mutation operation of evolutionary process to obtain an improved performance. As mentioned above, since the learning process is ex-
Table 4. Performance Comparison of Different Methods in Simulations

<table>
<thead>
<tr>
<th>Method</th>
<th>Regular</th>
<th>Up Peak</th>
<th>Down Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWT</td>
<td>LWR</td>
<td>AWT</td>
</tr>
<tr>
<td>GNP-RL with weight tuning</td>
<td>25.5</td>
<td>10.9</td>
<td>21.2</td>
</tr>
<tr>
<td>GNP-RL without weight tuning</td>
<td>26.7</td>
<td>11.8</td>
<td>21.9</td>
</tr>
<tr>
<td>Original GNP</td>
<td>30.0</td>
<td>14.8</td>
<td>22.1</td>
</tr>
<tr>
<td>AT Method</td>
<td>32.4</td>
<td>17.2</td>
<td>23.8</td>
</tr>
<tr>
<td>THV Method</td>
<td>30.2</td>
<td>15.6</td>
<td>20.6</td>
</tr>
</tbody>
</table>

Fig. 6. Performances under various Traffic Densities

(c) Down Peak Time

Fig. 6. Performances under various Traffic Densities

4.3.1 Evolutionary and Learning Results Under a Certain Traffic Density we simulate both GNP-RL without weight tuning and GNP-RL with weight tuning in a certain traffic mode †, respectively (Down Peak and Regular:2000 persons/hour, Up Peak:1800 persons/hour) by employing the proposed method with \( \varepsilon=0.1 \) greedy policy. Fig.5 shows the

† In this traffic mode, there are no idle cars existing, nor overload happening.
fitness curves of the proposed method in three traffic patterns comparing to the method using GNP without learning i.e., original GNP. Table 4 lists the performances of GNP-RL with weight tuning, GNP-RL without weight tuning and original GNP as well as two other conventional methods ("AT Method"~(14) – assigning to the elevator whose estimated arrival time is the shortest, “THV Method”~(15) – assigning to an elevator considering the distance between an elevator and the emerged hall call and its direction). In Fig.5, fitness curve of the GNP-RL without weight tuning in “Regular Time” demonstrates a rapidly converged result than without learning function (ε = 1.0), while a better and rapidly one in “Down Peak Time”. The figure shows that original GNP takes about 300 generations to converge while only 50 generations or less is needed to reach the same fitness level in GNP with RL with a lot of simulation time cut. That is to say, we can obtain the similar results comparing GNP-RL terminated at 50th generation (though it does not converge yet) and original GNP terminated at 300th generation. Note that the result in “Up Peak Time” is just commensurate with each other. The reason causing this result is due to the use of “compulsory dispatch strategy” in up peak mode. Since the traffic flow in “Up Peak Time” mainly consists of those passengers emerging at lobby and heading for upper floors, the compulsory dispatch strategy would be efficient enough in this case without employing any additional constraints such as setting a dispatching threshold of the number of loaded passengers at lobby~(16). This can also explain why we did not make an improvement of the performance, even a little bit worse in “Up Peak Time” of the proposed method, while a significant improvement was obtained in two other modes as shown in Table 4.

On the other hand, as shown in Fig.5, fitness curves of GNP-RL with weight tuning present a significant improvement during the evolutionary and learning process comparing to the method using GNP-RL without weight tuning. Also, we can confirm from Table 4 that both AWT and LWR in all three traffic patterns are decreased to some extent. These results show that it is useful to evolve each evaluation item considering its own “importance” than fixing it on an identical value when making car assignment decision.

### 4.3.2 Generalization Ability under Various Traffic Densities

We also made the GNP-RL without weight tuning under various traffic densities to check the generalization ability of the proposed method. The controller using GNP-RL and original GNP was constructed using the best individual obtained during the evolutionary and learning process in the above experiments. Traffic density varies from light to heavy. The transport capacity in different traffic modes is different, and generally it is the smallest in up-peak~(17). Fig.6 shows the results of the proposed method comparing to four other methods. In this figure, we cut the results when the traffic density is over 2400 persons/hour in regular and down-peak time and 2100 persons/hour in up-peak time, since the system becomes overloaded with several hundreds seconds waiting time in such a heavy traffic. The proposed method works well under a light to heavy traffic density in all traffic modes except the up-peak due to the reason discussed above.

Similarly, after checking the generalization ability of the best individual obtained in the GNP-RL without weight tuning, we also run the best individual from the GNP-RL with weight tuning under the same various traffic densities. As shown in Fig.6, the best individual evolved during the evolutionary process with tuning importance weight performs well as GNP-RL without weight tuning or better in some cases both on AWT and LWR. Note that the improvement of performances using GNP-RL with weight tuning is relatively small comparing to the method using GNP-RL without weight tuning even there exist big improvements shown in Fig.5. Considering that it also takes more time to converge in regular time during evolutionary process, weight tuning does not work as expected, and further studies are worth doing in the future. Moreover, it seems that there are some performance differences between the training and generalization process. This would be due to the definition of the fitness function which includes the term of “loop” while the generalization ability is just evaluated by the average passenger waiting time. This problem is a kind of dilemma since it is need to consider such a term of “loop” in the fitness function to eliminate those individuals with “loop”. We would study it in the future to find some solutions.

### 5. Conclusions and Future Study

In this paper, we proposed an elevator group controller using Genetic Network Programming (GNP) with Reinforcement Learning (RL). This method is motivated by the past research of applying GNP to EGSCS and applications of GNP with RL to virtual worlds. The proposed method was tested on our EGSCS simulator using traffic profiles of a typical office building. To verify its efficiency, we firstly run the evolutionary and learning process of the proposed method in a certain traffic mode and tested it comparing to another method using original GNP as well as two other conventional methods (AT, THV). Performance results show that the proposed method outperforms other three methods in all three traffic patterns except up peak with some reasons discussed above. Finally, we tested the proposed method under various traffic densities to check its generalization ability, and obtained a positive answer in point of the better performances. Moreover, to explore more performance improvement with the framework using GNP-RL, we made another experiment called GNP-RL with weight tuning by tuning the importance weight of the macro-processing nodes. The experiment results show that some further studies are needed for a significant improvement.

In the future, we plan to study further the importance weight tuning method proposed in this paper for more improvements, and search for some other sophisticated factors of the EGSCS. Also, the problem of performance differences between training process and generalization
test is another future work. Finally, we would like to apply the proposed method to the next generation elevator system, e.g., the double-deck elevator group system.

(Manuscript received Aug. 1, 2006, revised Feb. 16, 2007)

References


Jin Zhou (Non-member) From 1996 to 2004, he was with BBH Corp. He received ME degree from Waseda University, Japan in 2006. Since April, 2006, he has been a PhD student in Graduate School of Information, Production and System, Waseda University, Japan.

Lu Yu (Non-member) She received the BE degree from Beijing Institute of Technology, China in 2005. Since September, 2005, she has been a Master course student at Graduate School of Information, Production and System, Waseda University, Japan.

Shingo Mabu (Member) He received the BE and ME degrees in Electrical Engineering from Kyushu University, Japan in 2001 and 2003, respectively, and Ph. D degree from Waseda University, Japan in 2006. Since April 2006, he has been a Visiting Lecturer in Graduate School of Information, Production and Systems of Waseda University. Dr. Mabu is a member of the Society of Instrument and Control Engineers, the Institute of Electrical Engineers of Japan and IEEE.

Kotaro Hirasawa (Member) He received the BE and ME degrees from Kyushu University, Japan in 1964 and 1966, respectively. From 1966 to 1992, he was with Hitachi Ltd, where he served as a vice president of Hitachi Research Laboratory. From December 1992 to August 2002, he was a Professor in Graduate School of Information Science and Electrical Engineering of Kyushu University. Since September 2002, he has been a Professor in Graduate School of Information, Production and Systems of Waseda University. Dr. Hirasawa is a member of the Society of Instrument and Control Engineers, the Institute of Electrical Engineers of Japan and IEEE.

Jinglu Hu (Member) He received the MS degree in Electronic Engineering from Zhongshan University, China in 1986, and the Ph.D degree in Computer Science and Engineering from Kyushu Institute of Technology, Japan in 1997. From 1986 to 1993, he worked in Zhongshan University, where he was a Research Associate and the Lecturer. From 1997 to March 2003, he was a Research Associate in Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan. Since April 2003, he has been an Associate Professor in Graduate School of Information, Production and Systems of Waseda University. Dr. Hu is a member of the Society of Instrument and Control Engineers, the Institute of Electrical Engineers of Japan.

Sandor Markon (Non-member) He received Dipl.Ing. from Budapest Technical University in 1973. From 1975 to 1977, he was a research student at Kyoto University. Since 1979, he joined Fujitec Co.Ltd. He received PhD degree from Kyoto University in 1996. Dr. Markon is a member of IEEE, INNS, JNNS and SSIJ.