Double-Deck Elevator Systems using Genetic Network Programming with Reinforcement Learning

Jin Zhou, Lu Yu, Shingo Mabu, Kotaro Hirasawa, Jinglu Hu
Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, Japan
zhoujin@asagi.waseda.jp, yulu1006@fuji.waseda.jp, mabu@waseda.jp
hirasawa@waseda.jp, jinglu@waseda.jp

Sandor Markon
FUJITEC Co.Ltd. Prod.Dev.HQ,
Big Wing, Hikone, Shiga, Japan
markon@rd.fujitec.co.jp

Abstract—In order to increase the transportation capability of elevator group systems in high-rise buildings without adding elevator installation space, double-deck elevator system (DDES) is developed as one of the next generation elevator group systems. Artificial intelligence (AI) technologies have been employed to find some efficient solutions in the elevator group control systems during the late 20th century. Genetic Network Programming (GNP)[5], a new evolutionary computation method, has been reported to be employed as the elevator group system controller in some studies of recent years. Moreover, reinforcement learning (RL) is also verified to be useful for more improvements of elevator group performances when it is combined with GNP. In this paper, we proposed a new approach of DDES using GNP with RL, and did some experiments on a simulated elevator group system of a typical office building to check its efficiency. Simulation results show that the DDES using GNP with RL performs better than the one without RL in regular and down-peak time, while both of them outperforms a conventional approach and a heuristic approach in all three traffic patterns.

I. INTRODUCTION

The double-deck elevator system (DDES) was developed in 1930’s to increase the transportation capacity of elevator group systems in high-rise buildings, saving installation space by contrast with adding more elevators in single-deck elevator systems (SDES). It works most efficiently in up-peak traffic pattern, stopping at every other floor and serving two adjacent floors simultaneously. Until after the 1970’s, however, DDES was barely installed since it is hard to control in other traffic patterns like regular and down-peak, due to restrictions on elevator movement caused by the fixed connection of the two decks in each elevator shaft. Some more efficient approaches of DDES are needed for its wide application since there are about several hundreds of double-deck elevators installed in the world during the past several decades. So far, all the existing double-deck systems use the conventional full collective control system with up and down call buttons[1]. Recently, some studies based on DDES with destination floor guidance system (DFGS) are reported to explore new approaches.

As we know, the elevator group control system is a very large scale stochastic dynamic optimization problem. Artificial intelligence (AI) technologies, such as Genetic Algorithm (GA)[2], Fuzzy Logic[3], Neural Networks (NNs)[4], have been employed to find some efficient solutions in the elevator group control systems during the late 20th century. Genetic Network Programming (GNP)[5], a new evolutionary computation method, has been applied in this field in recent years, and its applicability and efficiency were clarified. Moreover, in our past studies, Reinforcement Learning (RL) was successfully introduced into the elevator group control systems using GNP[6]. Most of the approaches developed for the conventional elevator group systems (i.e., SDES), however, should be redesigned when applied in DDES due to the specific features of DDES. Since there are no good existing solutions for DDES yet, and also, the solution of using GNP with RL in SDES can not be directly applied to DDES which is more complex because of its specific features, we propose a new approach of DDES with DFGS using GNP and RL in this paper, and confirm its efficiency by doing simulations on a detailed elevator simulator. Simulation results are analyzed by comparing to other approaches.

This paper is organized as follows. DDES is introduced in the next section, and the proposed method is described in section III. Section IV shows the simulation results and some discussions. Finally, conclusions and future work are given in section V.

II. DOUBLE-DECK ELEVATOR SYSTEMS

A. Outline of DDES

Fig. 1. Outline of DDES
B. Running modes of DDES

There are three kinds of running modes in DDES. Generally speaking, these modes can be switched between each other if the hard systems and elevator group controller support to do so.

1) Double running mode: In this mode, the upper/lower cage serves only odd/even floors respectively. With about half of total stops cut, the transportation ability of DDES can be doubled in a pure up-peak traffic pattern. On the other hand, the movement of those passengers between odd and even floors would be restricted in such a running mode.

2) Semi-double running mode: Both the upper and lower cages can serve every floor except for the two lobby floors in this mode. Thus the DDES could provide a more flexible service for all possible traffic flows, though it makes the control algorithm more complex. The proposed method in this paper is based on this running mode.

3) Single running mode: This running mode can be employed with one deck being out of service in some particular cases, e.g., when one of the decks is under maintenance. That is to say, the DDES runs like a SDES in this mode.

C. Specific features of DDES

Compared with SDES, there are several specific features in DDES where two decks in each shaft are connected. These features, of course, are important and necessary to be considered when developing a control algorithm for DDES.

III. DDES USING GNP WITH RL

A. Overview of GNP with RL

Genetic Network Programming (GNP) has been proposed several years ago and also studied to be applied in the field of elevator group systems[5]. In this paper, reinforcement learning is introduced and combined with GNP to take advantage of the sophisticated diversified search ability of evolution and the intensified search and online learning abilities of reinforcement learning. The running process of GNP with RL is basically similar to the one of the original GNP. 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 in a sense after the genetic operators are executed.

A main difference from the original GNP is that the judgment/processing node is extended to a macro one to implement the learning process. Fig. 2 shows the basic structure of GNP with RL. In Fig. 2, $K_i$ represents the node
type, which is the same as original GNP. $ID_{ip}(1 \leq p \leq m_i)$ shows the identification number of the node function. Here $m_i$ of all nodes is set to 2, i.e., GNP can have the node function $ID_{1}$ or $ID_{2}$ in the node. $Q_{ip}$ 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 $ID_{ip}$, $d_{ip}$ is the time delay spent for judgment or processing. $C_{ip}^A, C_{ip}^B, ...$ show the node number of the next node $j$. $d_{ip}^A, d_{ip}^B, ...$ mean time delays spent for the transition from node $i$ to node $j$.

1) Evolutionary process: GNP with RL also has three kinds of genetic operators, i.e., selection, crossover and mutation. All of them except for mutation are the same as the original GNP. In GNP with RL, mutation operation could be executed not only on the connections among nodes but also on the function type and number of the sub-nodes in a macro-node. For simplicity, in this paper, we only execute the mutation operation on connections and the function type of sub-nodes.

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 $Q_{i1}, Q_{i2}, ..., Q_{im_i}$, and selects one of them based on $\epsilon$-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 $r_t$ and the next node $j$ becomes $C_{ip}^B$.

(3) At time $t+1$, GNP selects one $Q_{jp'}$ in the same way as step 1.

(4) Then the following procedure is executed.

$$\delta = r_t + \gamma Q_{jp'} - Q_{ip}$$

$$Q_{ip} \leftarrow Q_{ip} + \alpha \delta$$

(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.

B. DDES using GNP with RL

Figure 4 shows the outline of the proposed method. The elevator group system controller is implemented by a network structure, namely the GNP with RL. During the evolutionary and learning process, all individuals of GNP with RL in each generation are executed and evaluated in the DDES simulator. After that, the best individual of the last generation will be applied to DDES as the optimized controller.

There are four parts in the proposed controller, System Information Judgment Part, Candidate Cage Selection Part, Candidate Cage Confirmation Part, and Cage Assignment Part. Each part consists of some processing/judgment nodes described later. The controller is activated when a new hall call occurs, and evaluates all cages based on some predefined items, finally assigns the optimal cage to serve the new hall call. Since the upper and lower cages are evaluated respectively, we call the cage (either the upper cage or the lower one) “self cage” when it is under the current evaluation, and the other one “other cage”.

---

![Fig. 2. Basic Structure of GNP with RL](image1)

![Fig. 3. An example of node transition](image2)
1) Evaluation items: Based on some apriori knowledge of the elevator group control system, we define and employ 12 cage evaluation items to make the cage assignment decision. The first 6 evaluation items are common in SDES and DDES, which are indicated by the lower suffix \(sd\), and the remaining 6 ones are defined for DDES according to its specific features, indicated by the lower suffix \(d\).

- **\(AT_{sd}\): Predicted arrival time of the assigned hall call to the self cage including the incremental arriving time of the already registered hall calls to the self cage**
- **\(AET_{sd}\): Maximum of the arrival time plus elapsed time since assignment of the assigned hall calls to the self cage**
- **\(NP_{sd}\): Number of passengers in the self cage**
- **\(NC_{sd}\): Number of assigned hall calls to the self cage**
- **\(RR_{sd}\): Predicted riding rate (passenger number/cage capacity) of the self cage when the self cage arrives at the assigned hall call including the incremental riding rate of already registered hall calls to the self cage**
- **\(CHC_{sd}\): Check whether the new hall call coincides with the cage calls of the self cage**
- **\(AT_{d}\): Sum of the incremental predicted arrival time of the already assigned hall calls to the other cage**
- **\(AET_{d}\): Maximum of the arrival time plus elapsed time since assignment of the assigned hall calls to the other cage**
- **\(DNP_{d}\): Difference of the number of passengers between the self and other cage**
- **\(DNC_{d}\): Difference of the number of assigned hall calls between the self and other cage**
- **\(CCS_{d}\): Check the coincident service**
- **\(CSR_{d}\): Check the separate riding for identical destination**

Where, \(DNP_{d}\), \(DNC_{d}\) are defined considering the specific feature of DDES for unbalanced load. Similarly, \(CCS_{d}\) for one cage service and coincident service, \(CSR_{d}\) for separate riding for identical destination.

2) Main algorithm: As mentioned above, the controller using GNP with RL consists of four parts. The new hall call is classified in the System Information Judgment Part, considering its direction, origin/destination floor and the variance of elevators’ position. In Candidate Cage Selection Part, a candidate cage is selected from all cages based on 12 evaluation items. The candidate cage will be evaluated again by different evaluation items one by one in Candidate Cage Confirmation Part, and it will be assigned to the new hall call in Call Assignment Part if it is confirmed as the optimal one. Otherwise, another candidate cage is selected by returning to the Candidate Cage Selection Part.

**System Information Judgment Part**: In this part, the new hall call is classified based on three following terms, the degree of the variance of the elevator positions \(VP_{sd}\), the origin floor and direction of the new hall call \(EF_{sd}\) and the destination floor of the new hall call \(DF_{sd}\) \(VP_{sd}\) is used for the binary judgment whether the degree of the variance of the elevator positions is less than the average one over past 5 minutes or not. \(EF_{sd}\) is used for the judgment of the origin floor and direction of the new hall call with 5 branches, i.e., \{Base, General-Low-Up, General-Low-Down, General-High-Up, General-High-Down\}. \(DF_{sd}\) is used for the judgment of the destination floor of the new hall call with 3 branches, \{Base, General-Low, General-High\}.

**Candidate Cage Selection Part**: A candidate cage is selected in this part by the following equations. First, the cage evaluation function \(e(i)\) of cage \(i\) is calculated by Eq. (1).

\[
e(i) = \sum_{p \in P} w_X(p) \cdot x_p(i),
\]

where,

\(P\) : Set of suffixes of nodes transited in the cage selection part \((P\) is determined by node transition\)

\(w_X(p)\) : Weight of evaluation item \(X\) at the cage selection macro-node \(p\) \((w_X(p)\) is optimized during evolutionary process\)

\(x_p(i)\) : Normalized value of evaluation item \(X\) of cage \(i\) at the cage selection macro-node \(p\)

The normalized value \(x_p(i)\) is calculated by Eq. (2).

\[
x_p(i) = \frac{X_p(i)}{X_{AveMax}},
\]

where,

\(X_p(i)\) : Value of evaluation item \(X\) (corresponding to the selected sub-node) of cage \(i\) at the cage selection macro-node \(p\)

\(X_{AveMax}\) : Maximum value of average evaluation item \(X\) over past 5 minutes among cages.

The reason of using the normalized value of \(x_p(i)\) is that different evaluation items have different scales. As for the
evaluation item \( \{CHC_{sd}, CCS_{d}\} \), \( x_p(i) = 0 \) if satisfied, and \( x_p(i) = 1 \) if not satisfied. It is reversed in the case of \( \{CSR_{d}\} \). Finally, the candidate cage \( d \) is selected by Eq. (3).

\[
d = \arg \min_{i \in I} e(i),
\]

where, \( I \) : set of cage IDs

**Candidate Cage Confirmation Part** The selected candidate cage \( d \) is evaluated again by different evaluation items one by one to confirm whether it is the optimal one or not. In cage judgment nodes in this part, the binary judgment like Eq. (4) is carried out except for \( \{CHC_{sd}, CCS_{d}, CSR_{d}\} \).

\[
if \quad y_j(d) \leq r^Y_j, \quad j \in J
\]

where,

\( J \) : Set of suffixes of nodes in candidate cage confirmation part

\( y_j(d) \) : Normalized value of evaluation item \( Y \) of cage \( d \) at the cage judgment node \( j \)

\( r^Y_j \) : Threshold parameter of evaluation item \( Y \) of the cage judgment node \( j \) (\( r^Y_j \) is optimized during evolutionary process)

\( y_j(d) \) is also calculated by the following equation similar to Eq. (2).

\[
y_j(d) = \frac{Y_j(d)}{Y_{AveMax}},
\]

where,

\( Y_j(d) \) : Value of evaluation item \( Y \) of cage \( d \) at the cage judgment node \( j \)

\( Y_{AveMax} \) : Maximum value of averaged evaluation item \( Y \) over past 5 minutes among cages.

As for \( \{CHC_{sd}, CCS_{d}, CSR_{d}\} \), the binary judgment (satisfy/not) is done. If Eq. (4) is satisfied and cage judgment node \( j \) is connected to the node in the cage assignment part, then the new hall call is assigned to the optimal cage \( d \) in the cage assignment part. Otherwise, i.e., the candidate cage \( d \) does not satisfy Eq.(4), which means the condition of evaluation item \( Y \) is not satisfied, then, the node transition is resumed from the cage selection part in order to select another candidate cage again.

**Cage Assignment Part** The new hall call is assigned to the candidate cage by cage assignment nodes. Node transition returns to the system information judgment part after assignment, and the same procedures are executed for the next new hall call.

3) **Node functions:** The node functions in each part are defined as follows.

**System Information Judgment Node (3 kinds) : Judgment node**

- \( J^{VP}_{sd} \) : Judge the variance of the elevator position (2 branches).
- \( J^{DP}_{sd} \) : Judge the destination floor of the new hall call (3 branches).
- \( J^{EF}_{sd} \) : Judge the emerged floor and direction of the new hall call (5 branches).

**Cage Selection Node (12 kind) : Macro-processing node**

In each macro-processing node, there are two sub-nodes which are defined by one of the 12 evaluation items as follows. That is to say, there are totally 12 kinds of sub-nodes, and the types of sub-nodes in each macro-processing node are initialized randomly.

- \( S(X) \) : Select evaluation item \( X \) from 12 items by the node transition in the cage selection part and calculate Eq. (1).

\[
X \in \{AT_{sd}, AET_{sd}, NP_{sd}, NC_{sd}, RR_{sd}, CHC_{sd}, AT_d, AET_d, DNP_d, DNC_d, CCS_d, CSR_d\}
\]

**Cage Judgment Node (12 kinds) : Judgment node**

- \( J^Y(d) \) : Judge whether \( y_j(d) \leq r^Y_j \) is satisfied or not (2 branches)

\[
Y \in \{AT_{sd}, AET_{sd}, NP_{sd}, NC_{sd}, RR_{sd}, CHC_{sd}, AT_d, AET_d, DNP_d, DNC_d\}
\]

- \( J^Z(d) \) : Judge whether \( Z \) of cage \( d \) is satisfied or not (2 branches)

\[
Z \in \{CHC_{sd}, CCS_d, CSR_d\}
\]

**Cage Assignment Node (1 kind) : Processing node**

- \( A(d) \) : Assign cage \( d \) to the new hall call

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

\[
r = e^{-kT},
\]

where, \( T \) is the waiting time of the new hall call, and \( k \) is the coefficient of \( T \).

5) **Fitness function:** Fitness function shown in Eq. 7 is defined considering the following: (1) Minimization of waiting time of passengers; (2) Optimization of comfortable riding index; (3) Elimination of the loop gene of GNP.

\[
f = \frac{1}{N} \sum_{n=1}^{N} \left(t_n\right)^2 + w_1 \cdot \left(t_{max}\right)^2 + w_2 \cdot \left(Nc\right)^2 + w_3 \cdot l^2,
\]

where,

\( N \) : Total number of passengers

\( t_n \) : Waiting time of \( n \)-th passenger

\( t_{max} \) : Maximum waiting time among \( N \) passengers

\( Nc \) : Total number of passengers experiencing one cage service

\( l \) : Number of loops of GNP per one hour evaluation

\( w_1, w_2, w_3 \) : Weighting coefficients which are set by trial and error.

All terms in this function are expected to minimize due to its definitions described above. Thus, an individual with smaller fitness value means that it has a better structure and fitter parameters.

2007 IEEE Congress on Evolutionary Computation (CEC 2007) 2029
IV. Simulation Results and Discussions

A. DDES Simulator

In this paper, the proposed method is verified on a DDES simulator which simulates the DDES based on 0.1 second time slice. In each time slice, passenger events such as arriving at floor, pushing hall call button, boarding and exiting cage, are firstly determined according to a given O/D table shown in Table I (Row : origin floor, Column : destination floor), where three typical traffic patterns are defined. Here, the probability of passenger arrival is determined by exponential distribution from the start time of simulation considering the passenger density and the traffic flow ratio in Table I. Then elevator events such as stopping at floor, door open and close, travelling with nonlinearly changed speed, are implemented. Table II shows the specifications of the DDES simulator.

B. Running parameters

Table III shows running parameters of the proposed method, including the parameters of evolutionary process and those of reinforcement learning process.

C. Fitness curves

To optimize the DDES controller using GNP with or without RL, we run the evolutionary and learning process in three typical traffic patterns under a certain density shown in table II. Figure 5 shows the fitness curves of GNP with RL and without RL in three traffic patterns. The curves of GNP with RL converge to a lower level than the ones of GNP without RL in both down-peak time and regular time. That means the controller using GNP is optimized further when RL is introduced, since a lower fitness value denotes a better optimization result, such as smaller average waiting time, fewer number of passengers with one cage service, and so on, due to its definition described above. On the other hand, it should be noted that there is almost no improvement of the fitness curves in up-peak time. This case also occurred in our past study of SDES [6], because of the employing of the compulsory dispatch strategy.

D. Performance comparisons

After the evolutionary and learning process of the proposed method described above was terminated by a preset condition, such as 300 generations in this paper, the best individual is selected as the controller and applied to the DDES simulator for generalization tests. Since there is no explicitly published algorithm for DDES control yet, we employ two other DDES controllers as the conventional methods to verify the efficiency of the proposed method. One is the THV method [7], which is used in SDES. The other one is a heuristic control method for DDES, which assigns the cage with the smallest summation value of all 12 evaluation items used in the proposed method to the new hall call. Contrast to SDES, the passenger number with experiencing one cage service is usually considered as a performance criterion in DDES, besides those like AWT (average waiting time), AST (average system time, i.e. sum of average waiting time and average travel time) and LWR (long waiting ratio, namely the ratio of passengers who wait more than 60s).

As shown in Fig. 6, 7 and 8 where the results of all methods are the averages of 30 runs, either GNP with RL or GNP without RL has made a remarkable improvement over the other two conventional methods on all four performance criterions. Moreover, as the fitness curves in Fig. 5 show, GNP with RL also performed better than GNP without RL for all methods in up-peak time. This case also occurred on, due to its definition described above. On the other hand, it should be noted that there is almost no improvement of the fitness curves in up-peak time.
density point in the lower two figures of Fig. 8 because of overload of the system. The worst performances of the THV method suggests that a useful control algorithm of SDES may become unavailable in DDES.

result is obtained when introducing RL into GNP framework in DDES. Moreover, the proposed method outperformed over three other methods in the generalization tests. Since the selection of the evaluation items based on apriori knowledge is important to the design of the DDES controller, we plan to extract some more useful information of DDES and introduce them into our GNP framework in the future.

REFERENCES


V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new approach of DDES using GNP with RL and verified its efficiency on a detailed DDES simulator. The fitness curves show that a better optimization