Study on Genetic Network Programming-based Controllers of Elevator Group Systems

ZHOU, Jin

Graduate School of Information, Production and Systems
Waseda University

February 2009
# Contents

## 1 Introduction

1.1 Background ........................................ 1

1.2 Overview of Elevator Group Supervisory Control System (EGSCS) .... 2
  1.2.1 Development History of Elevator Group System ................. 3
  1.2.2 Existing Control Algorithms ............................. 4

1.3 Contents of this Research ............................. 6
  1.3.1 Genetic Network Programming (GNP) ....................... 6
  1.3.2 Objective and Motivation ................................. 7
  1.3.3 Research Topics ....................................... 8

## 2 EGSCS using GNP with Macro Nodes and RL  

2.1 Introduction ......................................... 10

2.2 EGSCS using GNP with Macro Nodes and RL .................. 11
  2.2.1 Outline of the Proposed Method .......................... 11
  2.2.2 Evaluation items ...................................... 12
  2.2.3 Node functions ....................................... 13
  2.2.4 Reward function ...................................... 15
  2.2.5 Fitness function ...................................... 15

2.3 Simulation Results and discussions ......................... 16
  2.3.1 Simulator specifications ................................. 16
  2.3.2 Execution conditions ................................... 17
  2.3.3 Evolutionary and learning results under a certain traffic density . 18
  2.3.4 Generalization ability under various traffic densities .......... 20

2.4 Conclusions and Future Study ............................ 21
3 DDES using GNP with Reinforcement Learning

3.1 Introduction ........................................ 24
3.2 Double-Deck Elevator Systems using GNP with RL .......................... 25
  3.2.1 Outline of the Proposed Method .................................. 25
  3.2.2 Evaluation items ............................................ 27
  3.2.3 Main algorithm ................................................ 28
  3.2.4 Node functions ............................................... 31
  3.2.5 Reward function .............................................. 32
  3.2.6 Fitness function ............................................... 33
3.3 Simulation Results and Discussions .................................... 33
  3.3.1 DDES Simulator ............................................. 33
  3.3.2 Running parameters .......................................... 34
  3.3.3 Fitness curves ................................................ 34
  3.3.4 Performance comparisons .................................... 35
  3.3.5 Space Saving .................................................. 37
3.4 Conclusions And Future Study ........................................ 38

4 Traffic flow-adaptive controller of DDES using GNP

4.1 Introduction .................................................. 41
4.2 Traffic flow of DDES ............................................ 42
4.3 Traffic flow-adaptive DDES controller using GNP .......................... 42
  4.3.1 Outline of the Proposed Method .................................. 42
  4.3.2 Evaluation items .............................................. 43
  4.3.3 Main algorithm ................................................ 45
  4.3.4 Node functions ............................................... 48
  4.3.5 Fitness function ............................................... 49
4.4 Simulation Results and Discussions .................................... 49
4.4.1 DDES Simulator ........................................... 49
4.4.2 Running parameters ....................................... 50
4.4.3 Fitness curves ............................................. 50
4.4.4 Performance comparisons .................................. 50
4.4.5 Study of the Obtained Control Rules ..................... 53
4.5 Conclusions And Future Study ............................ 53

5 Idle Cage Assignment-Embedded DDES using GNP 59
   5.1 Introduction .............................................. 59
   5.2 Idle Cage Assignment-embedded DDES controller using GNP .... 59
      5.2.1 Outline of the Proposed Method ......................... 59
      5.2.2 Evaluation Items ..................................... 60
      5.2.3 Main Algorithm ....................................... 62
      5.2.4 Node Functions ....................................... 65
      5.2.5 Fitness Function ..................................... 65
   5.3 Simulation Results and Discussions ......................... 66
      5.3.1 DDES Simulator ....................................... 66
      5.3.2 Fitness curves ........................................ 67
      5.3.3 Performance comparisons ................................ 67
   5.4 Conclusions ................................................ 71

6 Conclusions .................................................. 72

References ..................................................... 74
# A Genetic Network Programming (GNP)

A.1 Basic Structure of GNP .................................................. 81
A.2 Genotype and Phenotype of GNP ...................................... 84
A.3 Genetic Operation of GNP ............................................... 86

## A.3.1 Selection .......................................................... 86
## A.3.2 Crossover ......................................................... 87
## A.3.3 Mutation ......................................................... 89

# B Genetic Network Programming with RL

B.1 Structure of GNP with RL ............................................. 92
B.2 Running Process of GNP with RL ................................... 94

## B.2.1 Evolutionary process ............................................. 94
## B.2.2 Learning process ................................................ 95

# C Elevator Group Supervisory Control System (EGSCS)

C.1 Outline of EGSCS ..................................................... 98
C.2 Cage Assignment Policies in EGSCS ................................ 98
C.3 Traffic Patterns in EGSCS ............................................ 100
C.4 Performance Criteria of EGSCS .................................... 100

# D Double-Deck Elevator Systems (DDES)

D.1 Outline of DDES ....................................................... 102
D.2 Running modes of DDES ............................................. 103
D.3 Specific features of DDES .......................................... 104
D.4 Cage assignment policy ............................................. 105
D.5 Destination Floor Guidance System .............................. 106

# E Framework of Switching Method

E.1 Fitness Curves of Switching Method ............................... 108
E.2 Switching Results of Switching Method .......................... 109
Acknowledgments 110

Research Achievements 111
Chapter 1

Introduction

1.1 Background

Artificial Intelligence (AI) has been proposed in the middle of the 20th century, attempting to create machines with “intelligence” such as reasoning, knowledge, planning, learning, communication, perception and the ability to move and manipulate the objects that human beings have acquired during their life. With the rapid development of computers in the past several decades, AI achieved its greatest successes in the 1990s and early 21st century throughout a wide range of fields including medical diagnosis, stock trading, robot control, law, scientific discovery, etc. So far, many approaches in the field of AI, such as evolutionary computation, Neural Networks (NNs) and Fuzzy Logics (FL), have been proposed and studied collectively by the emerging discipline of computational intelligence.

Among them, there is evolutionary computation which mainly comprises Genetic Algorithms (GA), Evolutionary Programming (EP), Evolution Strategy (ES), Genetic Programming (GP) and learning classifier systems. The mechanisms of evolutionary computation were inspired by biological evolution such as reproduction, mutation, recombination, natural selection and survival of the fittest. In this approach, the best solution for the problem is selected after the iterative progress (so-called evolutionary progress) where all candidate solutions compete with each other generation by generation (for more details, see http://en.wikipedia.org/wiki/Evolutionary_computation). Up to now,
many studies have been reported for applying evolutionary computation to optimization problems in some complicated systems. With its efficiency and effectiveness verified by experiments, evolutionary computation has been adopted in more and more real system developments.

On the other hand, elevator group system has been widely studied as a large-scale stochastic dynamic optimization problem since the appearance of the high-rise buildings. Unlike some other transportation systems like train systems which run on a pre-scheduled time table, the elevator group system should make decisions on-line based on the stochastic and incomplete information on the traffic situation in buildings. Due to its vast state space, significant uncertainty and numerous resource constraints, it is hard to manage elevator group systems using conventional control methods. With the advent of the booming research on AI technologies in the late 20th century, many AI-based approaches in this field have been proposed and, as a result, lots of elevator companies have released their new productions using these approaches. To meet the continuously increased demands for higher performances of elevator group systems and space saving of its installation, developments of the elevator systems keep going both on the hardware configurations and the control algorithms. Meanwhile, the progress in the research of AI technologies makes it possible and valuable to apply the latest proposal to the field of the elevator group system. Nowadays, the utilization of AI and expert systems in the context of the elevator system control is conceivably an indispensable trend [1].

1.2 Overview of Elevator Group Supervisory Control System (EGSCS)

After the invention of elevator system about 150 years ago, elevator technologies have been developed mostly in two ways, i.e., the hardware configurations and the control algorithms, which are commonly affected and driven by each other. In this research, some new control algorithms based on the aforementioned Genetic Network Programming (GNP) are proposed considering various hardware configurations of the elevator systems.
1.2. Overview of Elevator Group Supervisory Control System (EGSCS)

1.2.1 Development History of Elevator Group System

Elevator group system has been proposed to manage more than two elevators which run as a group in buildings. To provide more efficient service with a given specification of the elevator system installation site, the elevator group system has been developed ranging from the Single-Deck Elevator System (SDES) to Double-Deck Elevator System (DDES) and Multi-Car Elevator System (MCES).

As shown in Fig. 1.1, many control algorithms have been proposed and adopted in this domain, using the early relay-based collective approach and the late microprocessor-based dynamic programming and AI technologies. On the other hand, there are some other proposals for better performances by introducing some new hardware configurations such as camera monitoring system [2] and destination floor guidance system. Note that the practical control algorithms are seldom independent to the hardware configuration even
though some control theory could be applicable to all the systems without considering
the difference of hardware configuration. Therefore, there exist many elevator systems
developed or under developing based on various combinations of control algorithms and
hardware configurations.

1.2.2 Existing Control Algorithms

Some collective approaches including [3, 4] have been proposed and commonly adopted
at an early stage of elevator group systems development. One of them is known as AT [5]
approach, in which the new hall call is generally served by the car with the shortest
estimated arrival time from its current position to the floor where the new hall call
emerged. Another approach is called THV [6], in which not only the distance between
the current position of the candidate cars and the new hall call floor, the trip direction
of the candidate cars is also considered in the fitness function. The pseudocode of THV
is listed as following.

**THV pseudocode**

N = number of floors in the building
Read the system current state
d = Distance (call, car) = |call floor - elevator floor|
IF elevator is homing to the call floor with the same trip direction of the hall call
   Fitness Function = N + 1 -d
ELSE IF
elevator is homing to the call floor with trip direction different from the hall call
   Fitness Function = N - d
ELSE IF elevator has just leaved the floor of the hall call
   Fitness Function = 1
ELSE (the elevator is stopped)
   Fitness Function = N - d
Car Allocation = Best Fitness Function

With the advent of mega high-rise buildings, zoning and dynamic zoning approaches
are proposed to provide desired performances in the elevator group systems. On the other
1.2. Overview of Elevator Group Supervisory Control System (EGSCS)

hand, the drastic development in the computer industry provided a good chance with the powerful computing abilities for developing more sophisticated elevator group controller. Approaches based on dynamic programming or markov chain, where a huge search space should be handled, have been proposed though they were designated under a moderately simplified search space for some specific traffic pattern.

The development of elevator group systems, in the late 20th century, benefited from the advancement of AI technologies, and many AI-based elevator group systems have been proposed. Typically, Genetic Algorithms (GA), Evolutionary Strategy (ES), Fuzzy Logic (FL) and Neural Networks (NNs) have been employed.

In the approaches using genetic algorithm [7–11], some researchers proposed to encode the weight vectors of car evaluation function as an individual of GA and to optimize them during the evolutionary process. After that, the car evaluation function with the optimized weight vectors was used to make car assignment decision when a new hall emerges. Contrastively, some others proposed to encode the assignment solution as an individual of GA, in which all hall and car calls are served considering usage of all cars. Once the evolutionary process ends, all calls including the new hall call are registered to cars’ service lists according to decoding result of the best solution.

In the approaches using fuzzy logic [12–17], the inference mechanisms of fuzzy logic are usually employed not only to identify the traffic patterns of buildings but also to make the car assignment decisions based on some preset expert control rules. In practice, such approaches can be smoothly embedded into the existing expert control systems without any strong impact on the system control.

In the approaches using neural networks [18–22], similar to some aforementioned approaches using fuzzy logic, the neural networks could be trained on a set of simulated situations in advance to identify the current traffic pattern out of several predefined ones. Also, the neural networks could be trained to make the car assignment decision according to the signal of output layer. In such cases, the states of elevator group systems possibly
Table 1.1 Released AI-based Elevator Group Systems

<table>
<thead>
<tr>
<th>Prod. Name</th>
<th>Maker</th>
<th>Date</th>
<th>Adopted AI Tech., Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS9000</td>
<td>KONE</td>
<td>1991</td>
<td>Fuzzy, for traffic pattern forecast</td>
</tr>
<tr>
<td>ΣAI-2200</td>
<td>MITSUBISI</td>
<td>2000</td>
<td>NNs, for traffic pattern forecast</td>
</tr>
<tr>
<td>AI-2100</td>
<td>1998</td>
<td></td>
<td>Fuzzy, for cage assignment decision making</td>
</tr>
<tr>
<td>FI-340G</td>
<td>HITACHI</td>
<td>1996</td>
<td>GA, for elevator service program generating</td>
</tr>
<tr>
<td>COMMAND-AI</td>
<td>TOSHIBA</td>
<td>1989</td>
<td>Fuzzy, for cage arrival time estimate</td>
</tr>
</tbody>
</table>

up to several dozens, including the position and direction of each car, the load of each car, the hall/car halls location, and so on, are commonly introduced into the neural networks as the input layer. The neural network-based controller can be trained by given some teacher signals (from existing controllers) at an early stage, and trained further by reinforcement learning for more flexible control [23].

In some other approaches, more than one AI technologies are combined such as evolutionary strategies with neural networks, fuzzy logic with neural networks, etc. All of studies on AI-based elevator group systems verified the efficiency and effectiveness of AI technologies in this domain. As a result, many productions using AI technologies listed in Table 1.1 have been released by several well known elevator companies.

1.3 Contents of this Research

1.3.1 Genetic Network Programming (GNP)

Genetic Algorithm (GA) [24] and Genetic Programming (GP) [25,26] have been studied well as the typical evolutionary computation methods in many fields during past decades. A lot of papers based on them have verified their applicability and availability. On the other hand, however, some of their limitation and weakness emerged with further research. Therefore, many extended algorithms have been proposed to enhance them based on their original architecture. Genetic Network Programming (GNP) [27] is one of them. Comparing to string encoded structure of GA and tree encoded structure of GP, GNP is proposed as a network encoded structure to improve the expression ability. Moreover,
due to the directed graph network structure, the judgment nodes and processing nodes can be repeatedly used in GNP, so the gene of it becomes compact causing the efficient evolution of GNP. Also, during the transition of the nodes in GNP the past route can influence the current judgment and processing which contributes to the modeling of Non Markov Processes. To verify those features of GNP mentioned above, many studies have been done and the applicability of GNP to both virtual-world problem [28] like tile-world models and real-world problem [29] like elevator group control system has been clarified. The details of GNP are described in App. A.

1.3.2 Objective and Motivation

In the early studies, some basic algorithms for applying GNP to elevator group systems have been proposed. Improvement of performances over the collective approaches such as $AT$, $THV$ shows the applicability of GNP in this domain. On the other hand, reinforcement learning has been introduced into the framework of GNP for faster training speed and better performances. The efficiency and effectiveness of this approach called GNP with Reinforcement Learning (RL) have been verified by simulation results on some virtual-world problems. As further studies, GNP with RL is firstly proposed to apply to the elevator group system with single deck in each shaft called Single-Deck Elevator Systems (SDES) for better performances. This approach is then extended to Double-Deck Elevator System (DDES) considering the specific features of the system.

All simulations, in the former studies, are based on three typical traffic patterns of office buildings for simplicity. However, since the traffic pattern usually keeps changing during a day, a traffic flow-adaptive controller for DDES using GNP is proposed as a sequential study. Finally, the attention is focused on the light traffic mode of DDES where some idle cages emerge, and an idle cage assignment algorithm embedded controller for DDES is proposed.
1.3.3 Research Topics

In this dissertation, there are four research topics to be studied based on the aforementioned objective and motivation. As mentioned early, although there are many approaches proposed till now with reports of favorable results, at present, these are not comparable and not reproducible. The reason is that the elevator systems studied by each researcher are all different, with many varying parameters (number of cars, number of floors, traffic conditions etc.), proprietary control algorithms, etc [30]. Therefore, all the proposals of this research are verified on a detailed simulator of elevator group systems, comparing with some collective approaches and other heuristic approaches.

In chapter 2, GNP with RL is introduced into Elevator Group Supervisory Control System (EGSCS) for faster training and better performances during the task execution period of each individual. There are six sub-nodes in macro-nodes, with each sub-node defined by one of six evaluation items. The reinforcement learning process is based on the \( \varepsilon \)-greedy policy and a reward function defined by passenger waiting time. To verify the efficiency of the proposed method, fitness curves and test results are compared with the method using original GNP on three typical traffic patterns (i.e., up-peak time, regular and down-peak time). Moreover, performances comparison is also made with other two collective methods (\( AT \), \( THV \)) under varying traffic densities from 300 persons per hour to more than 2100.

In chapter 3, with the applicability and availability of applying GNP with RL to single-deck elevator group systems verified in the last chapter, GNP with RL is proposed to enhance the Double-Deck Elevator System (DDES) using GNP. Unlike what is imagined, the algorithm of using GNP with RL in single-deck elevator group systems can not be directly applied to DDES which is more complex because of its specific features. A new algorithm is developed for DDES, and its efficiency is verified by comparing with a heuristic method and \( THV \) method as well as the one using GNP without RL. In addition,
the effectiveness of the space saving using GNP with RL is clarified based on comparisons with single-deck elevator systems.

In chapter 4, a DDES controller using GNP is proposed to adapt the varying traffic flow during a work day in the buildings. By contrast with the approaches in the former studies, a traffic identification part is introduced into the DDES controller using GNP in which cage assignment function is to be localized based on a typical varying traffic flow after the evolutionary process ends. Another approach optimized by GA, in which the controller is switched between three GNP controllers optimized under three traffic patterns respectively, is developed to compete with the proposed method. The better performances of the proposed method over the switching method and other two methods verify the effectiveness of the proposal. To analyze the control rules of the proposed method, the gene of one of the best individuals is studied as well as the judgment result based on the individual in the test process.

In chapter 5, in order to improve the performances of DDES in the light traffic mode, the DDES controller using GNP studied in the previous chapter is extended to embed idle cage assignment based on the cage idling event and timer. When a cage becomes idle or the timer triggers a time-up, all floors are evaluated as the destination for the dispatch of the idle cage to. Three items of the elevator group systems are proposed to make the evaluation. To verify the efficiency of the proposal, the performances are compared with the method without idle cage assignment as well as several other heuristic methods including the methods with fixed dispatchment to some certain floors. Some experiments are made under varying passenger densities from 200 persons per hour to 2700, and the simulation results confirm that the proposed method works efficiently in the light traffic mode improving the performances very little as supposed in advance.

In chapter 6, after giving the objectives and motivation of each research topic in this thesis, some conclusions about the proposed algorithms are drawn based on the simulation results.
Chapter 2

EGSCS using GNP with Macro Nodes and RL

2.1 Introduction

Since Genetic Network Programming (GNP) has been proposed as a new evolutionary computation method several years ago, it was firstly applied to Elevator Group Supervisory Control System (EGSCS) in a real world problem [29]. 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) [28, 31] 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 chapter, 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 chapter, 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.

2.2 EGSCS using GNP with Macro Nodes and RL

2.2.1 Outline of the Proposed Method

Figure 2.1 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).
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 chapter, 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.
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.

2.2.3 Node functions

In this chapter, 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. (2.1) 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 chapter, 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.

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

(2.1)

where, $L$ : set of suffixes of cars,

$J$ : set of suffixes of passed macro-processing nodes during node transition of $P_1$,

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

$w_j$ : importance weight of macro-processing node $j$,

$d$ : a new hall call is assigned to car number $d$.

- **Judgment Node**
2.2. EGSCS using GNP with Macro Nodes and RL

$J_1$: 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*.

### 2.2.4 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.

$$r = e^{-kT}, \quad (2.2)$$

where, $T$ is the waiting time of the new hall call, and $k$ is the coefficient of $T$.

### 2.2.5 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. (2.3). 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.

\[
f = \frac{1}{|P|} \sum_{p \in P} (t_p)^2 + w_t \times (t_{max})^2 + w_n \times (n)^2,
\]

(2.3)

where \( |P| \) is the total number of passengers during the simulation time, \( t_p \) is the waiting time of passenger \( p \in P \), \( t_{max} \) is the maximum waiting time, \( n \) is the number of the loops, and \( w_t \) and \( w_n \) are the coefficient of \( t_{max} \) and \( n \), respectively.

2.3 Simulation 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 \textit{GNP-RL without weight tuning} by fixing the importance weight at 1.0 for each macro-processing node. And after that, the \textit{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 executed when a new hall call occurs, each individual will “learn” about 1300 times in one generation without any additional node transition time except for some computational costs for Q value update.

2.3.1 Simulator specifications

Table 2.1 shows the specifications of an EGSCS simulator which is created as our testbed. In this chapter, 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 2.2, 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.
2.3. Simulation Results and discussions

Table 2.1 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 2.2 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></td>
<td>TF</td>
<td>GF</td>
<td>TF</td>
</tr>
<tr>
<td>Terminal Floor(TF)</td>
<td>–</td>
<td>5</td>
<td>–</td>
</tr>
<tr>
<td>General Floor(GF)</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2.3.2 Execution conditions

Parameters of the proposed method are listed in Table 2.3. Note that the size of generation and population is set at 500 and 310 respectively in some of our past studies [32]. 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 chapter, 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.
Table 2.3 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>◦ Macro-Processing Node $P_1$</td>
<td>10(10/Kind)</td>
</tr>
<tr>
<td>◦ Processing Node $P_2$</td>
<td>10(10/Kind)</td>
</tr>
<tr>
<td>◦ Judgment Node $J_1$</td>
<td>10(10/Kind)</td>
</tr>
<tr>
<td>◦ Boot Node</td>
<td>1</td>
</tr>
<tr>
<td>Mutation Probability $P_m$</td>
<td>0.01</td>
</tr>
<tr>
<td>Crossover Probability $P_c$</td>
<td>0.1</td>
</tr>
<tr>
<td>Learning Coefficient $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Discount Ratio $\gamma$</td>
<td>0.7</td>
</tr>
<tr>
<td>$\varepsilon$-greedy $\varepsilon$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

2.3.3 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\(^1\), 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.2.2 shows the fitness curves of the proposed method in three traffic patterns comparing to the method using GNP without learning i.e., original GNP. Table 2.4 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” – assigning to the elevator whose estimated arrival time is the shortest, “THV Method” [6] – assigning to an elevator considering the distance between an elevator and the emerged hall call and its direction). In Fig.2.2 fitness curve of the *GNP-RL without weight tuning* in “Regular Time” demonstrates a rapidly converged result than without learning function ($\varepsilon=1.0$), while a

\(^1\)In this traffic mode, there are no idle cars existing, nor overload happening.
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
Table 2.4 Performance Comparison of Different Methods in Simulations

<table>
<thead>
<tr>
<th>Method</th>
<th>Regular</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AWT</td>
<td>LWR</td>
<td>AWT</td>
<td>LWR</td>
<td>AWT</td>
<td>LWR</td>
</tr>
<tr>
<td>GNP-RL with weight tuning</td>
<td>25.5</td>
<td>10.9</td>
<td>21.2</td>
<td>6.5</td>
<td>29.9</td>
<td>12.1</td>
</tr>
<tr>
<td>GNP-RL without weight tuning</td>
<td>26.7</td>
<td>11.8</td>
<td>21.9</td>
<td>7.2</td>
<td>30.8</td>
<td>13.4</td>
</tr>
<tr>
<td>Original GNP</td>
<td>30.0</td>
<td>14.8</td>
<td>22.1</td>
<td>7.4</td>
<td>36.0</td>
<td>19.0</td>
</tr>
<tr>
<td>AT Method</td>
<td>32.4</td>
<td>17.2</td>
<td>23.8</td>
<td>8.7</td>
<td>38.2</td>
<td>21.4</td>
</tr>
<tr>
<td>THV Method</td>
<td>30.2</td>
<td>15.6</td>
<td>20.6</td>
<td>6.6</td>
<td>45.8</td>
<td>15.6</td>
</tr>
</tbody>
</table>

in this case without employing any additional constraints such as setting a dispatching threshold of the number of loaded passengers at lobby [33]. 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 2.4.

On the other hand, as shown in Fig.2.2, 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 2.4 that both AWT (the average waiting time of all passengers over the simulation period) and LWR (the ratio of passengers who wait more than 60s) 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.

### 2.3.4 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 [34]. Fig.2.3 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.2.3, 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.2.2. 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.

2.4 Conclusions and Future Study

In this chapter, we proposed an elevator group controller using Genetic Network Programming (GNP) with Reinforcement Learning (RL). This method is motivated by the
Fig. 2.3 Performances under various Traffic Densities
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 chapter 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.
Chapter 3

DDES using GNP with Reinforcement Learning

3.1 Introduction

The double-deck elevator system (DDES) was developed in 1930's to increase the transportation capacity of elevator group control systems in high-rise buildings, saving the installation space without 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 movements caused by the fixed connection of the two cages in each elevator shaft. Some more efficient approaches of DDES are needed for its wide application, though 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. Recently, some studies based on DDES with destination floor guidance system (DFGS) are reported to explore new approaches. For more details of DDES, see App. D.

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), Fuzzy Logic, Neural Networks (NNs), have been employed to find some efficient solutions in the elevator group control systems during the late 20th century. Genetic
Network Programming (GNP), a new evolutionary computation method, has been applied to this field in recent years, and its applicability and efficiency were clarified. Moreover, in our past studies, Reinforcement Learning (RL) was successfully introduced to the elevator group control systems using GNP [35], in which a special implementation of RL has been made for elevator systems since the way of implementing RL is different problem by problem. Most of the approaches developed for the conventional elevator group systems (i.e., SDES), however, should be redesigned when applied to DDES due to the specific features of DDES. Since there are no good exiting 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 chapter, and confirm its efficiency by doing simulations on a detailed elevator simulator. Simulation results are analyzed comparing to other approaches.

3.2 Double-Deck Elevator Systems using GNP with RL

3.2.1 Outline of the Proposed Method

Figure 3.1 shows the outline of the proposed method. The elevator group system controller is implemented by a network structure, namely the GNP with RL. In this chapter, the controller of DDES is evolved offline on a DDES simulator firstly. During the evolutionary and learning process, all individuals of GNP with RL in each generation are executed and evaluated in the DDES simulator. We set the evaluation time of an individual of GNP at 2 hours. It takes about 4 days to complete all the evolutionary and learning process (about 180,000 simulated hours) on an Pentium 2.1GHz desktop computer with Linux OS installed. After that, the best individual of the last generation will be applied to the real DDES (the same DDES simulator in our case) as the optimized controller.
Since the control rules and parameters are represented in the best individual, the online computation time of the controller only depends on the speed of the microprocessor, and thus can be very small.

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 evaluation items determined by GNP structure, 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”.

### 3.2.2 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 \(s_d\), and the remaining 6 ones are defined for DDES according to its specific features, indicated by the lower suffix \(d\).

\(AT_{s_d}\): 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

\(AET_{s_d}\): Maximum of the predicted arrival time of the assigned hall call and predicted arrival time plus elapsed time of the already registered hall calls since their assignment to the self cage

\(NP_{s_d}\): Number of passengers in the self cage

\(NC_{s_d}\): Number of assigned hall calls to and cage calls in the self cage

\(RR_{s_d}\): 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

\(CHC_{s_d}\): 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 predicted arrival time plus elapsed time of the already registered hall calls since their assignment 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 and cage 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. These evaluation items are used to determine the candidate cage for a new hall call. It suggests that the cage with less unbalanced load, less one cage service, less separate riding for identical destination and more coincident service will tend to be assigned. Moreover, the weights of these evaluation items are optimized during the evolutionary process to make a proper balance between them.

### 3.2.3 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 $V_{Ps_d}$, the origin floor and direction of the new hall call $E_{Fs_d}$ and the destination floor of the new hall call $D_{Fs_d}$. $V_{Ps_d}$ 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. $E_{Fs_d}$ is used for the judgment of the origin floor and direction of the new hall call with 5 branches, i.e., \{Lobby Floor, Low General Floor with Up Direction, Low General Floor with Down Direction, High General Floor with Up Direction, High General Floor with Down Direction\}. $D_{Fs_d}$ is used for the judgment of the destination floor of the new hall call with 3 branches, \{Lobby Floor, Low General Floor, High General Floor\}.

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. (3.1).

\[
e(i) = \sum_{p \in P} w_p \cdot x_p(i), \tag{3.1}
\]

where,

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

$w_p$: Weight of the cage selection macro-node $p$ ($w_p$ is optimized during evolutionary process)

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

The normalized value $x_p(i)$ is calculated by Eq. (3.2).

\[
x_p(i) = \frac{X_p(i)}{X_{AveMax}}, \tag{3.2}
\]

where,
\( X_p(i) \): Value of evaluation item \( X \) (corresponding to the selected sub-node) of cage \( i \) at 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. (3.3).

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

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. (3.4) is carried out except for \( \{CHC_{sd}, CCS_d, CSR_d\} \).

\[
y_j(d) \leq r_j^Y, \quad j \in J
\]

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 cage judgment node \( j \)

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

\( y_j(d) \) is also calculated by the following equation similar to Eq. (3.2).

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

where,
3.2. Double-Deck Elevator Systems using GNP with RL

\( Y_j(d) \) : Value of evaluation item \( Y \) of cage \( d \) at 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. (3.4) 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. (3.4), which means the condition of evaluation item \( Y \) is not satisfied, then, the node transition is resumed from the candidate 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.

3.2.4 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).
- \( J^{DF}_{sd} \) : Judge the destination floor of the new hall call (3 branches).
- \( J^{EF}_{sd} \) : Judge the origin floor and direction of the new hall call (5 branches).

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

In each macro-processing node, there are two sub-nodes, each of which is 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 candidate cage selection part and calculate Eq. (3.1).

\[
X \in \{ AT_{sd}, AET_{sd}, NP_{sd}, NC_{sd}, RR_{sd}, CHC_{sd}, AT_{d}, AET_{d}, DNP_{d}, DNC_{d}, CCS_{d}, CSR_{d} \}
\]

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

• **J^Y(d)** : Judge whether \( y_j(d) \leq r_j^Y \) is satisfied or not (2 branches)

\[
Y \in \{ AT_{sd}, AET_{sd}, NP_{sd}, NC_{sd}, RR_{sd}, AT_{d}, AET_{d}, DNP_{d}, DNC_{d} \}
\]

• **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

### 3.2.5 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 punishment (a small reward value).

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

where, \( t \) is the waiting time of the new hall call, and \( k \) is the coefficient of \( t \).
3.2.6 Fitness function

Fitness function shown in Eq. 3.7 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.

\[ f = \frac{1}{N} \sum_{n=1}^{N} \left( t_n \right)^2 + w_t \cdot \left( t_{\text{max}} \right)^2 + w_c \cdot \left( N_c \right)^2 + w_l \cdot \left( N_l \right)^2, \]  

(3.7)

where,

\[ N : \text{Total number of passengers} \]
\[ t_n : \text{Waiting time of } n-\text{th passenger} \]
\[ t_{\text{max}} : \text{Maximum waiting time among } N \text{ passengers} \]
\[ N_c : \text{Total number of passengers experiencing one cage service} \]
\[ N_l : \text{Number of loops of GNP per one hour evaluation} \]
\[ w_t, w_c, w_l : \text{Weighting coefficients which are set by trial and error.} \]

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.

3.3 Simulation Results and Discussions

3.3.1 DDES Simulator

In this chapter, 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 buttons, boarding and exiting cages, are determined according to a given O/D table shown in Table 3.1 (Row : origin floor, Column : destination floor), where three typical traffic patterns are defined. Then elevator events such as stopping at floor, door opening and closing, travelling with nonlinearly changed speed, are implemented. Table 3.2 shows the specifications of the DDES simulator.
<table>
<thead>
<tr>
<th>Items</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Floors</td>
<td>16</td>
</tr>
<tr>
<td>Number of Shafts (Cages)</td>
<td>6 (12)</td>
</tr>
<tr>
<td>Floor Distance [m]</td>
<td>4.5</td>
</tr>
<tr>
<td>Max. Velocity [m/s]</td>
<td>2.5</td>
</tr>
<tr>
<td>Max. Acceleration [m/s²]</td>
<td>0.7</td>
</tr>
<tr>
<td>Jerk [m/s³]</td>
<td>0.7</td>
</tr>
<tr>
<td>Cage Capacity [person]</td>
<td>20</td>
</tr>
<tr>
<td>Time for Opening Door [s]</td>
<td>2.0</td>
</tr>
<tr>
<td>Time for Closing Door [s]</td>
<td>2.3</td>
</tr>
<tr>
<td>Time for Riding [s/person]</td>
<td>1.0</td>
</tr>
<tr>
<td>Passenger Density [person/h]</td>
<td></td>
</tr>
<tr>
<td>— Regular Time</td>
<td>3000</td>
</tr>
<tr>
<td>— Up-peak Time</td>
<td>2700</td>
</tr>
<tr>
<td>— Down-peak Time</td>
<td>3300</td>
</tr>
</tbody>
</table>

### 3.3.2 Running parameters

Table 3.3 shows running parameters of the proposed method, including the parameters of evolutionary process and those of reinforcement learning process.

### 3.3.3 Fitness curves

To optimize the DDES controller using GNP with or without RL \(^1\), we run the evolutionary and learning process in three typical traffic patterns \(^2\) under a certain density

---

\(^1\)To make an optimal balance between exploration and exploitation in the reinforcement learning process, the \(\varepsilon\)-greedy policy is employed in this research, in which exploration is weighted when \(\varepsilon\) is given a large value while exploitation is weighted when \(\varepsilon\) is given a small value. Here, \(\varepsilon\) is set at 0.1 and 1.0 for employing the learning process or not, respectively.

\(^2\)We employ the compulsory dispatch strategy for all methods in up-peak traffic pattern, by which the idle cages will be compulsorily dispatched to the lobby floor.
3.3. Simulation Results and Discussions

<table>
<thead>
<tr>
<th>Items</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>—Crossover</td>
<td>120</td>
</tr>
<tr>
<td>—Mutation</td>
<td>170</td>
</tr>
<tr>
<td>—Elite</td>
<td>10</td>
</tr>
<tr>
<td>Node Size</td>
<td>91+Initial Node</td>
</tr>
<tr>
<td>—$ J^{VP_{d}}, J^{DF_{d}}, J^{EF_{d}}$</td>
<td>9 (3/kind)</td>
</tr>
<tr>
<td>—$ S(X)$</td>
<td>60 (5/kind)</td>
</tr>
<tr>
<td>—$ J^Y(d), J^Z(d) $</td>
<td>12 (1/kind)</td>
</tr>
<tr>
<td>—$ A(d) $</td>
<td>10 (10/kind)</td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>0.1</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.01</td>
</tr>
<tr>
<td>Learning Coefficient $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Discount Ratio $\gamma$</td>
<td>0.7</td>
</tr>
<tr>
<td>Evaluation Time [hour]</td>
<td>2</td>
</tr>
<tr>
<td>$w_t, w_c, w_f, k$</td>
<td>0.007, 0.001, 0.6, 0.007</td>
</tr>
</tbody>
</table>

shown in Table 3.2. Figure 3.2 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 [35], because of the employing the compulsory dispatch strategy.

3.3.4 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 research, the best in-
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 [6], which is used in SDES. The other one is a heuristic control method for DDES, which assigns the new hall call to the cage with the smallest summation value of all 12 evaluation items used in the proposed method. Contrast to SDES, the passenger number experiencing one cage service is usually considered as a performance criterion in DDES, besides AWT (average waiting time), AST (average...
system time, i.e. sum of average waiting time and average traveling time) and \( LWR \) (long waiting ratio, namely the ratio of passengers who wait more than 60s).

As shown in Fig. 3.3, 3.4, and 3.5, 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. 3.2 show, GNP with RL also performed better than GNP without RL in down-peak and regular time. It should be noticed that the curves of the \( THV \) method stop around at the 2300 passengers’ traffic density point in Fig. 3.5 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.

### 3.3.5 Space Saving

As mentioned above in the introduction part, DDES was invented to increase the transportation capacity of elevator group systems without adding more shaft spaces, which usually means a lot of costs especially in high-rise buildings. In this subsection, we did some experiments to verify the effect of space saving by comparing the \( AWT \) of DDES and SDES. For a convincing comparison, SDES was simulated with the same specifications in Table 3.2 defined for DDES, except only one cage in each shaft \(^3\). “AT Method” \([5]\), in which the elevator with the shortest estimated arrival time is assigned to the new hall call, was employed as the controller against the proposed method. The system having \( x \) shafts with \( y \) cages in each shaft is denoted by \( SxCy \). For instance, the SDES having 10 shafts, with 1 cage in each shaft, of course, is denoted by \( S10C1 \).

Fig. 3.6 shows the results of \( AWT \) of SDES using \( AT \) and DDES using GNP-RL and \( AT \) in three traffic patterns. From this figure, space saving can be verified by the performances of DDES and SDES in the range of 30 ~ 40 [s] of \( AWT \). The equivalence of DDES to SDES is shown as follows. (a) down-peak time: \( S6C2 \) (GNP-RL) \( \rightarrow \) \( S9C1 \),

\(^3\)The space of one shaft in DDES is the same as the one in SDES.
S6C2 (AT) → S8C1, (b) regular time: S6C2 (GNP-RL) → S9C1, S6C2 (AT) → S8C1, (c) up-peak time: S6C2 (GNP-RL) → S12C1, S6C2 (AT) → S10C1, showing that DDES could save spaces about 30[%] ∼ 50[%], and more in those cases when passengers’ traffic density is higher. The figure also shows that S6C2 (GNP-RL) works better than S6C2 (AT) with saving spaces equivalent to 1 ∼ 2 shafts in SDES.

### 3.4 Conclusions And Future Study

In this chapter, we proposed a new approach for DDES using GNP with RL considering the specific features of DDES. The efficiency and effectiveness of the proposed method have been verified based on a detailed DDES simulator. The fitness curves show that a better optimization 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 DDES controller using GNP with RL is proposed to apply to one of the three typical traffic patterns after optimized under the corresponding traffic pattern, a traffic-adaptive controller is very valuable to be studied considering the real application situation. This is exactly the research topic of the next chapter.
3.4. Conclusions And Future Study

![Fig. 3.3 Performance Comparisons in Down-peak Time](image1)

![Fig. 3.4 Performance Comparisons in Regular Time](image2)
Fig. 3.5 Performance Comparisons in Up-peak Time

(a) Down-peak  
(b) Time Regular Time

(c) Up-peak Time

Fig. 3.6 Space Saving of DDES compared to SDES
Chapter 4

Traffic flow-adaptive controller of DDES using GNP

4.1 Introduction

Since Genetic Network Programming was applied to elevator group systems including single-deck elevator system and double-deck elevator system, many studies [36, 37] have been done to explore more efficient cage assignment strategies based on several fixed traffic patterns, say, up-peak time, regular time and down-peak time in general. The cage assignment strategies obtained in these studies need to be switched manually when they are applied to the real elevator systems, since almost all of the traffic flows vary continuously during a day. Another study [38] shows us an approach in the single-deck elevator system (SDES), where the evolved cage assignment strategies are to be switched automatically depending on different traffic patterns. Even though the effectiveness of this proposal has been verified by the improved performances, some further studies are needed to explore a more sophisticated solution since the traffic flow is roughly divided into six parts in advance while it actually varies continuously. As a further study, in this chapter we proposed a controller of DDES adaptive to traffic flows by introducing a traffic flow judgment part into the Genetic Network Programming (GNP) framework. As compared with the former switching approach, the different parts of GNP are expected to work in a functionally localized way by the evolutionary process to make the appropriate cage assignment in different traffic flows. To verify the effectiveness of the proposed method, we did some experiments based on a detail-simulated DDES, comparing the proposed method with a conventional approach and two heuristic ones.
4.2 Traffic flow of DDES

Generally, almost all the traffic flows in many kinds of buildings vary dynamically during a whole day. In addition, the traffic flow patterns could be very different according to the type of buildings such as office building, residential building, and so on. Fig. 4.1 shows a traffic flow pattern in a typical office building during a work day, where four main periods are defined as incoming time (8:00-9:15), business time (9:15-11:45, 13:15-17:45), lunch time (11:45-13:15) and outgoing time (17:45-19:00).

4.3 Traffic flow-adaptive DDES controller using GNP

4.3.1 Outline of the Proposed Method

To adapt the varying traffic flow, a new traffic flow judgment part is introduced into the GNP controller of DDES in this chapter as shown in Fig. 4.2. Since the cage assignment
rules are different among various traffic flows, the candidate cage selection part should be functionally localized by the evolutionary process to make appropriate cage assignments in different traffic flows.

There are three parts in the proposed controller, *Traffic Flow Judgment Part, Candidate Cage Selection and Confirmation Part*, and *Cage Assignment Part*. Each part consists of some judgment/processing 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 lower one) “self cage” when it is under the current evaluation, and the other one “other cage”.

### 4.3.2 Evaluation items

Based on some a priori knowledge of the elevator group control system, we define and employ 12 cage evaluation items to make the cage assignment. 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, which are 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 predicted arrival time of the assigned hall call and predicted arrival time plus elapsed time of the already registered hall calls since their assignment to the self cage
- $NP_{sd}$: Number of passengers in the self cage
- $NC_{sd}$: Number of assigned hall calls to the self cage
Fig. 4.2 Outline of the Proposed Method
4.3. Traffic flow-adaptive DDES controller using GNP

$RR_{sd}$: 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 predicted arrival time plus elapsed time of the already registered hall calls since their assignment 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 unbalanced load in DDES. Similarly, $CCS_{d}$ for one cage service and coincident service, $CSR_{d}$ for separate riding for identical destination.

4.3.3 Main algorithm

When a new passenger arrives at a certain floor and pushes one button of DFGS, a signal of the new hall call will be sent to the GNP controller of DDES. Firstly, the Traffic Flow Judgment Part is activated to identify the current traffic flow based on the traffic flow information of the past 5 minutes. After that, the node transition is transferred to Candidate Cage Selection and Confirmation Part, where a candidate cage is selected from all cages based on the above 12 evaluation items. If this candidate cage fails to
meet the judgment condition of some evaluation items during the node transition, it will be discarded and another one is to be selected. This process is repeated until (1) the cage assignment is determined or (2) a loop is detected when the accumulated value of the time delays becomes larger than the preset threshold. The cage assignment will be determined by using another conventional cage assignment algorithm in the latter case. Finally, the cage is assigned to the new hall call by sending a control signal to DDES in Cage Assignment Part.

Traffic Flow Judgment Part In this part, three indexes of the traffic flow are employed to identify the current traffic flow. The first one is the ratio of the number of passengers departing from the base floor and the number of passengers arriving at the base floor during the past 5 minutes, denoted by $RBF$. The second one is the ratio of the number of passengers moving upward and the number of passengers moving downward during the past 5 minutes, denoted by $RUD$. The last one is the total number of passengers emerged during the past 5 minutes, denoted by $TNP$. As shown in Fig. 4.2, each branch of the judgment nodes in this part activates different node transition routes causing different node transition routes in the Candidate Selection and Confirmation Part depending on the traffic.

Candidate Cage Selection and Confirmation Part To determine the cage for the new hall call, there are two steps in this part. First, a candidate cage is selected by the cage evaluation function $e(i)$ defined by Eq. (4.1).

$$e(i) = \sum_{p \in P} w_p \cdot x_p(i), \quad (4.1)$$

where,

$P$ : Set of suffixes of nodes transited in the candidate cage selection part ($P$ is determined by evolution)
4.3. Traffic flow-adaptive DDES controller using GNP

\( w_p : \) Weight at the candidate cage evaluation node \( p \) (\( w_p \) is also optimized during the evolutionary process)

\( x_p(i) : \) Normalized value\(^1\) of evaluation item \( X \) of cage \( i \) at the candidate cage evaluation node \( p \)

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 \( \{CSR_d\} \).

After the candidate cage \( d \) is selected by \( d = \arg \min_{i \in I} e(i) \), where \( I \) is set of cage IDs, it is evaluated again by the evaluation items to confirm whether it is really appropriate or not. Candidate cage confirmation nodes are proposed here to play the confirmation. Except for the candidate cage confirmation nodes defined by \( \{CHC_{sd}, CCS_d, CSR_d\} \), the judgment \( y_j(d) \leq r_Y^j \) is made in the candidate cage confirmation nodes, where \( j \in J \), \( J \) is the set of suffixes of nodes in the candidate cage confirmation part, \( y_j(d) \) is the normalized value of evaluation item \( Y \) of cage \( d \) at the candidate cage confirmation node \( j \), \( r_Y^j \) is the threshold parameter of the evaluation item \( Y \) of the candidate cage confirmation node \( j \) (\( r_Y^j \) is also optimized during the evolutionary process). If the judgment result is NO, the current candidate cage will be discarded and returns to the candidate cage evaluation nodes for selecting another one. Otherwise, the confirmation process is continued by other candidate cage confirmation nodes or moves to the cage assignment part described next. As for \( \{CHC_{sd}, CCS_d, CSR_d\} \), the binary judgment (satisfy/not) is done.

**Cage Assignment Part**  After the candidate cage is determined in the former part, it is assigned here to serve the current new hall call. The cage assignment node will send a control signal to DDES and start the node transition after the next new hall call is detected.

---

\(^1\)Since different evaluation items have different scales, the normalized value \( x_p(i) \) is used and defined as follows. \( x_p(i) = \frac{X_p(i)}{X_{AveMax}} \), where \( X_p(i) \): value of evaluation item \( X \) of cage \( i \) at the node \( p \) in the candidate cage selection part, \( X_{AveMax} \): maximum value of average evaluation item \( X \) over past 5 minutes among cages.
4.3.4 Node functions

The node functions in each part are defined as follows.

Traffic Flow Judgment Node (3 kinds) : Judgment node

- \( J^{RBF} \) : Judge the ratio of the number of passengers departing from the base floor and the number of passengers arriving at the base floor during the past 5 minutes (5 branches).

- \( J^{RU} \) : Judge the ratio of the number of passengers moving upward and the number of passengers moving downward during the past 5 minutes (5 branches).

- \( J^{TN} \) : Judge the total number of passengers emerged during the past 5 minutes (5 branches).

Candidate Cage Evaluation Node (12 kind) : Processing node

Each processing node is defined by one of the following 12 evaluation items.

- \( S(X) \) : Select evaluation item \( X \) from 12 items by the node transition in the candidate cage selection part and calculate Eq. (4.1).

\[
X \in \{AT_{sd}, AET_{sd}, NP_{sd}, NC_{sd}, RR_{sd}, CHC_{sd}, AT_{d}, AET_{d}, DNP_{d}, DNC_{d}, CCS_{d}, CSR_{d}\}
\]

Candidate Cage Confirmation Node (12 kinds) : Judgment node

- \( J^{Y}(d) \) : Judge whether \( y_j(d) \leq r_Y^j \) is satisfied or not (2 branches)

\[
Y \in \{AT_{sd}, AET_{sd}, NP_{sd}, NC_{sd}, RR_{sd}, AT_{d}, AET_{d}, DNP_{d}, DNC_{d}\}
\]

- \( J^{Z}(d) \) : 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
4.3.5 Fitness function

Fitness function of Eq. (4.2) is used considering the following: (1) Minimization of the waiting time of passengers; (2) Optimization of comfortable riding index; (3) Elimination of the loop gene of GNP.

\[ f = \frac{1}{N} \sum_{n=1}^{N} (t_n)^2 + w_t \cdot (t_{\text{max}})^2 + w_c \cdot (N_c)^2 + w_l \cdot (N_l)^2, \]  

(4.2)

where,

\( N \) : Total number of passengers

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

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

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

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

\( w_t, w_c, w_l \) : Weight coefficients which are set by trial and error.

All terms in this function are expected to minimize due to its definitions. Thus, an individual with smaller fitness value has better structures.

4.4 Simulation Results and Discussions

4.4.1 DDES Simulator

The DDES simulator was built based on the specifications shown in Table 4.1. All events are simulated in detail by using 0.1 second time slice. In each time slice, the events of passengers such as arriving at floors, pushing the button of DFGS, getting on and off the cage, are generated according to the O/D table determined by the traffic flow shown in Fig. 4.1. Moreover, the passenger arrival is generated based on exponential distribution in this research. Then events of elevators such as stopping at floors, door open and close, travelling with nonlinearly changed speed, are also implemented referring to real DDES.
Table 4.1 Specifications of the DDES Simulator.

<table>
<thead>
<tr>
<th>Items</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Floors</td>
<td>16</td>
</tr>
<tr>
<td>Number of Shafts (Cages)</td>
<td>6 (12)</td>
</tr>
<tr>
<td>Floor Distance [m]</td>
<td>4.5</td>
</tr>
<tr>
<td>Max. Velocity [m/s]</td>
<td>2.5</td>
</tr>
<tr>
<td>Max. Acceleration [m/s²]</td>
<td>0.7</td>
</tr>
<tr>
<td>Jerk [m/s³]</td>
<td>0.7</td>
</tr>
<tr>
<td>Cage Capacity [person]</td>
<td>20</td>
</tr>
<tr>
<td>Time for Opening Door [s]</td>
<td>2.0</td>
</tr>
<tr>
<td>Time for Closing Door [s]</td>
<td>2.3</td>
</tr>
<tr>
<td>Time for Riding [s/person]</td>
<td>1.0</td>
</tr>
</tbody>
</table>

4.4.2 Running parameters

Table 4.2 shows the running parameters of the proposed method.

4.4.3 Fitness curves

Fig. 4.3 shows the best fitness curves of the proposed method during the evolutionary process. To reduce the influence of random noises, we did the experiments using 5 random seeds. Fitness curves of all 5 random seeds are described in Fig. 4.3 as well as the average one. Note that the population of GNP controller was optimized generation by generation and converged to a certain level at the latter generations of the evolutionary process.

4.4.4 Performance comparisons

To verify the effectiveness of the proposed method, we use a conventional method called *AT Method* and two other heuristic methods called *SUM Method* and *Switching Method* (see details in App. A), respectively, since there has not been published an algorithm yet for DDES control. In *AT Method*, the cage which arrives first at the floor is assigned, where the new hall call occurs. In *SUM Method*, the cage is assigned to the new hall

---

2To shorten the simulation test period, here the traffic data are sampled in every 5 minutes.
4.4. Simulation Results and Discussions

Table 4.2 Running Parameters of GNP.

<table>
<thead>
<tr>
<th>Items</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>— Crossover</td>
<td>120</td>
</tr>
<tr>
<td>— Mutation</td>
<td>170</td>
</tr>
<tr>
<td>— Elite</td>
<td>10</td>
</tr>
<tr>
<td>Node Size</td>
<td>106+Start Node</td>
</tr>
<tr>
<td>— $J_{RBF}$, $J_{RUD}$, $J_{TNP}$</td>
<td>24 (8/kind)</td>
</tr>
<tr>
<td>— $S(X)$</td>
<td>60 (5/kind)</td>
</tr>
<tr>
<td>— $J^V(d)$, $J^Z(d)$</td>
<td>12 (1/kind)</td>
</tr>
<tr>
<td>— $A(d)$</td>
<td>10 (10/kind)</td>
</tr>
<tr>
<td>Time Delay</td>
<td></td>
</tr>
<tr>
<td>— $J_{RBF}$, $J_{RUD}$, $J_{TNP}$</td>
<td>1</td>
</tr>
<tr>
<td>— $S(X)$</td>
<td>2</td>
</tr>
<tr>
<td>— $J^V(d)$, $J^Z(d)$</td>
<td>1</td>
</tr>
<tr>
<td>— $A(d)$</td>
<td>5</td>
</tr>
<tr>
<td>Time Delay Threshold Value</td>
<td>30</td>
</tr>
<tr>
<td>Crossover Rate</td>
<td>0.1</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.01</td>
</tr>
<tr>
<td>Evaluation Time [h]</td>
<td>2.2</td>
</tr>
<tr>
<td>$w_t$, $w_c$, $w_l$</td>
<td>0.007, 0.001, 0.6</td>
</tr>
</tbody>
</table>

call, which has the smallest sum of all 12 evaluation items used in the proposed method. Therefore, the evaluation function is defined as $e(i) = \sum_{x \in X} x(i)$, which is similar to Eq. (4.1). In *Switching Method*, the GNP controllers which are optimized to three fixed traffic patterns (i.e., up-peak time, regular time and down-peak time) are switched according to the ratio of the number of passengers moving upward and downward. Real-coded Genetic Algorithm is employed to optimize the parameters in this heuristic method for fair comparison with other methods. Furthermore, non-uniform mutation [39] is used to make a more precise tuning of the parameters.

Table 4.3 shows the performance comparison of the proposed method with other methods mentioned above. Two main performance criteria ($AWT$, $LWR$) are employed to evaluate the DDES controller implemented by different methods in this chapter, though
there are others such as energy consumption. \textit{AWT} represents the average waiting time of all passengers over the simulation period, and \textit{LWR} denotes the long waiting ratio, i.e., the ratio of passengers who wait more than 60s. In Table 4.3, conventional \textit{AT method} gives the worst performances among all methods, while, \textit{SUM method}, one of the heuristic methods, improves the performances to a remarkable level showing that the evaluation items used in the proposed method are important in DDES. The further improved performances of the \textit{switching method} show that it is useful to optimize the GNP controller even in three fixed traffic flow patterns. The best performances are given to the proposed method suggesting that the traffic flow judgment part and the functional localization realized by it are very important in the proposed method.
### Table 4.3 Performances Comparison of Different Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>LWR[%]</th>
<th>IMP.</th>
<th>AWT[s]</th>
<th>IMP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed method</td>
<td>7.5</td>
<td>36.3%</td>
<td>27.1</td>
<td>7.4%</td>
</tr>
<tr>
<td>switching method</td>
<td>8.4</td>
<td>28.5%</td>
<td>28.3</td>
<td>3.3%</td>
</tr>
<tr>
<td>SUM method</td>
<td>8.6</td>
<td>27.3%</td>
<td>28.1</td>
<td>4.1%</td>
</tr>
<tr>
<td>AT method</td>
<td>11.8</td>
<td>0.0%</td>
<td>29.3</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Note: Imp. is defined by \( \frac{\text{Value}_{\text{worst}} - \text{Value}_m}{\text{Value}_{\text{worst}}} \), \( \text{Value}_{\text{worst}} = \max_m \{ \text{Value}_m \} \), \( m \in \{ \text{proposed method, switching method, AT method, SUM method} \} \).

#### 4.4.5 Study of the Obtained Control Rules

To make a further study of the proposed method, we analyzed the evolved structure of the best individual and listed its rules for candidate cage selection in Table 4.4, 4.5 and 4.6. There are many different rules which are used to select the candidate cage according to different traffic flow patterns (classified by judgment nodes \( J_{RBF} \), \( J_{RUD} \) and \( J_{TNP} \)). A candidate cage is determined by Eq. (4.1) based on the evaluation items with the weight in brackets. Note that the evaluation items \( CHC_{sd} \), \( AT_{sd} \) and \( CCS_d \) are commonly used in most of the rules, suggesting that coincidence of the new hall call with the cage calls of self cage \( (CHC_{sd}) \), the predicted arrival time to self cage \( (AT_{sd}) \), and the coincident service of both two cages \( (CCS_d) \) are very important to select the candidate cage as well as their optimized weights. Fig. 4.4 shows the traffic flow judgment results during the test period, where the best GNP individual evolved under one of the 5 rand seeds was applied to DDES as the controller. The corresponding candidate cage selection rules based on Table. 4.4, 4.5 and 4.6 are used in Fig. 4.4.

#### 4.5 Conclusions And Future Study

In this chapter, we proposed a new DDES controller using GNP which can adapt to the varying traffic flows by introducing the traffic flow judgment part. In order to make...
Fig. 4.4 An Example of the Traffic Flow Judgment Results
a flexible cage assignment for the varying traffic flows, the *cage selection part* has become functionally localized by the evolutionary process in the proposed method. The efficiency and effectiveness of the proposed method have been verified on a detailed DDES simulator. Actually, it has been cleared that the proposed method outperformed over three other methods on two main performance criteria of DDES. In the next chapter, the attention of research is focused on the performance improvement in light traffic mode of DDES since the overall performances of elevator group systems are generally evaluated not only in heavy traffic mode but also in light traffic mode.
Table 4.4 Examples of Rules For Candidate Cage Selection.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>$J^{RBF}$</th>
<th>$J^{RUD}$</th>
<th>$J^{TNP}$</th>
<th>Cage Selection Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PTN_1$</td>
<td>$&lt;0.1$</td>
<td>$&lt;0.1$</td>
<td>$&lt;30$</td>
<td>$AET_d(0.824) \rightarrow CHC_{sd}(0.339) \rightarrow DNP_d(0.019) \rightarrow NC_{sd}(0.453) \rightarrow AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_2$</td>
<td>$&lt;40$</td>
<td></td>
<td></td>
<td>$CCS_d(0.452) \rightarrow NC_{sd}(0.453) \rightarrow AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_3$</td>
<td>$&lt;50$</td>
<td></td>
<td></td>
<td>$AT_d(0.900) \rightarrow CHC_{sd}(0.129) \rightarrow AT_{sd}(0.465)$</td>
</tr>
<tr>
<td>$PTN_4$</td>
<td>$&lt;60$</td>
<td></td>
<td></td>
<td>$AET_d(0.784) \rightarrow DNP_d(0.610) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_5$</td>
<td>$\geq60$</td>
<td></td>
<td></td>
<td>Same as $PTN_2$</td>
</tr>
<tr>
<td>$PTN_6$</td>
<td>$&lt;0.5$</td>
<td>$&lt;30$</td>
<td></td>
<td>$CHC_{sd}(0.339) \rightarrow DNP_d(0.019) \rightarrow NC_{sd}(0.453) \rightarrow AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_7$</td>
<td>$&lt;40$</td>
<td></td>
<td></td>
<td>$AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_8$</td>
<td>$&lt;50$</td>
<td></td>
<td></td>
<td>$CSR_d(0.411) \rightarrow NC_{sd}(0.761) \rightarrow DNP_d(0.610) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_9$</td>
<td>$&lt;60$</td>
<td></td>
<td></td>
<td>$AT_{sd}(0.339) \rightarrow CCS_d(0.452) \rightarrow NC_{sd}(0.453) \rightarrow AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_{10}$</td>
<td>$\geq60$</td>
<td></td>
<td></td>
<td>$AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_{11}$</td>
<td>$&lt;2.0$</td>
<td>$&lt;30$</td>
<td></td>
<td>$AT_{sd}(0.234) \rightarrow NP_{sd}(0.244) \rightarrow CHC_{sd}(0.339) \rightarrow DNP_d(0.019) \rightarrow NC_{sd}(0.453) \rightarrow AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_{12}$</td>
<td>$&lt;40$</td>
<td></td>
<td></td>
<td>$AET_d(0.327)$</td>
</tr>
<tr>
<td>$PTN_{13}$</td>
<td>$&lt;50$</td>
<td></td>
<td></td>
<td>$AET_d(0.017) \rightarrow CCS_d(0.452) \rightarrow NC_{sd}(0.453) \rightarrow AET_{sd}(0.688) \rightarrow AT_d(0.844) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_{14}$</td>
<td>$&lt;60$</td>
<td></td>
<td></td>
<td>$DNC_d(0.005)$</td>
</tr>
<tr>
<td>$PTN_{15}$</td>
<td>$\geq60$</td>
<td></td>
<td></td>
<td>$AET_d(0.876) \rightarrow CHC_{sd}(0.281) \rightarrow AT_{sd}(0.932) \rightarrow CCS_{sd}(0.630)$</td>
</tr>
<tr>
<td>$PTN_{16}$</td>
<td>$&lt;8.0$</td>
<td></td>
<td>$&lt;30$</td>
<td>$CSR_d(0.661) \rightarrow RR_{sd}(0.124) \rightarrow CSR_d(0.824) \rightarrow NP_{sd}(0.352) \rightarrow CHC_{sd}(0.657) \rightarrow NP_{sd}(0.129) \rightarrow DNP_d(0.645)$</td>
</tr>
<tr>
<td>$PTN_{17}$</td>
<td>$&lt;40$</td>
<td></td>
<td></td>
<td>Same as $PTN_{11}$</td>
</tr>
<tr>
<td>Pattern</td>
<td>$J^{RBF}$</td>
<td>$J^{RUD}$</td>
<td>$J^{TNP}$</td>
<td>Cage Selection Rules</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>$PTN_{18}$</td>
<td>$&lt;50$</td>
<td>$NP_{sd}(0.244)$ $CHC_{sd}(0.339)$ $DNP_d(0.019)$ $NC_{sd}(0.453)$ $AET_{sd}(0.688)$ $AT_d(0.844)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{19}$</td>
<td>$&lt;60$</td>
<td>$DNC_d(0.924)$ $NC_{sd}(0.761)$ $DNP_d(0.610)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{20}$</td>
<td>$\geq 60$</td>
<td>$NP_{sd}(0.352)$ $NP_{sd}(0.244)$ $CHC_{sd}(0.339)$ $DNP_d(0.019)$ $NC_{sd}(0.453)$ $AET_{sd}(0.688)$ $AT_d(0.844)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{21}$</td>
<td>$\geq 8.0$</td>
<td>Same as $PTN_{6} - PTN_{10}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{22}$</td>
<td>$0.1 \leq$</td>
<td>$&lt;0.1$</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
</tr>
<tr>
<td>$PTN_{23}$</td>
<td>$&lt;0.5$</td>
<td>$&lt;0.5$</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
</tr>
<tr>
<td>$PTN_{24}$</td>
<td>$&lt;2.0$</td>
<td>Same as $PTN_{6} - PTN_{10}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{25}$</td>
<td>$&lt;8.0$</td>
<td>$&lt;30$</td>
<td>Same as $PTN_{2}$</td>
<td></td>
</tr>
<tr>
<td>$PTN_{26}$</td>
<td>$&lt;40$</td>
<td>$CHC_{sd}(0.129)$ $AT_d(0.465)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{27}$</td>
<td>$&lt;50$</td>
<td>$DNC_d(0.617)$ $CHC_{sd}(0.129)$ $AT_d(0.465)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{28}$</td>
<td>$&lt;60$</td>
<td>$DNP_d(0.572)$ $DNC_d(0.005)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{29}$</td>
<td>$\geq 60$</td>
<td>$AET_{sd}(0.236)$ $CSR_d(0.824)$ $NP_{sd}(0.352)$ $CHC_{sd}(0.657)$ $NP_{sd}(0.129)$ $DNP_d(0.645)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{30}$</td>
<td>$\geq 8.0$</td>
<td>$&lt;30$</td>
<td>$NP_{sd}(0.129)$ $DNP_d(0.645)$</td>
<td></td>
</tr>
<tr>
<td>$PTN_{31}$</td>
<td>$&lt;40$</td>
<td>$AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{32}$</td>
<td>$&lt;50$</td>
<td>$DNP_d(0.610)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{33}$</td>
<td>$&lt;60$</td>
<td>$DNP_d(0.421)$ $AET_{sd}(0.561)$ $AET_d(0.017)$ $CCS_d(0.452)$ $NC_{sd}(0.453)$ $AET_d(0.688)$ $AT_d(0.844)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{34}$</td>
<td>$\geq 60$</td>
<td>Same as $PTN_{28}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{35}$</td>
<td>$0.5 \leq$</td>
<td>$&lt;0.1$</td>
<td>Same as $PTN_{6} - PTN_{10}$</td>
<td></td>
</tr>
<tr>
<td>$PTN_{36}$</td>
<td>$&lt;2.0$</td>
<td>$&lt;0.5$</td>
<td>$&lt;30$</td>
<td>$AET_{sd}(0.561)$ $AET_d(0.017)$ $CCS_d(0.452)$ $NC_{sd}(0.453)$ $AET_d(0.688)$ $AT_d(0.844)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
</tr>
<tr>
<td>$PTN_{37}$</td>
<td>$&lt;40$</td>
<td>$NC_{sd}(0.453)$ $AET_{sd}(0.688)$ $AT_d(0.844)$ $CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{38}$</td>
<td>$&lt;50$</td>
<td>Same as $PTN_{26}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{39}$</td>
<td>$&lt;60$</td>
<td>$CHC_{sd}(0.281)$ $AT_d(0.932)$ $CCS_d(0.630)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{40}$</td>
<td>$\geq 60$</td>
<td>Same as $PTN_{12}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pattern</td>
<td>$J_{RBF}$</td>
<td>$J_{RUD}$</td>
<td>$J_{TNP}$</td>
<td>Cage Selection Rules</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td>$PTN_{41}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{18}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{42}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{29}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{43}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{8}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{44}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{28}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{45}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{32}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{46}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{25} - PTN_{29}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{47}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{16} - PTN_{20}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{48}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{49}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{50}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{51}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{52}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{53}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{54}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{55}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{56}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PTN_{57}$</td>
<td>2.0 ≤ 50</td>
<td>Same as $PTN_{11} - PTN_{15}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: $PTN_i$ represents the $i$th pattern, and the different patterns having the same transition routes are marked by 'Same as $PTN_i$' or 'Same as $PTN_i - PTN_j$'.
Chapter 5

Idle Cage Assignment-Embedded DDES using GNP

5.1 Introduction

So far, most of the studies on DDES control algorithms [40, 41] including our past research [36, 37], have focused on the improvement of DDES performances in a heavy traffic mode, where all cages almost always keep moving to assigned calls. However, an overall evaluation of DDES is to be made in not only a heavy but also light traffic mode. Since some cages become idle in a light traffic mode, how to dispatch these idle cages, which is seldom considered in the heavy traffic mode, becomes important when developing the controller of DDES. In this chapter, we propose a DDES controller with idle cage assignment algorithm embedded using Genetic Network Programming (GNP) for a light traffic mode. Some experiments are done to verify the efficiency of the proposed method using a DDES simulator.

5.2 Idle Cage Assignment-embedded DDES controller using GNP

5.2.1 Outline of the Proposed Method

In our past studies, a GNP controller of DDES has been proposed for the semi-double running mode, and its performance has been verified under a moderately heavy traffic density, where all cages always keep moving to assigned calls. The traffic density, however,
does not keep high during the work days in typical high rise office buildings. Some cages become idle when the DDES runs in a light traffic mode, and they are usually requested to stay there until they are assigned to a new hall call. An idle cage assignment algorithm of how to dispatch these idle cages is proposed in this chapter for some performance improvements.

Since how to dispatch the idle cages is important for a DDES controller in a light traffic mode, we added an floor assignment algorithm to the DDES controller using Genetic Network Programming (GNP). Fig. 5.1 shows the outline of the proposed method, where the GNP controller consists of two parts, i.e. Cage Assignment algorithm (CA) and Floor Assignment algorithm (FA). The immediate assignment policy is employed in Cage Assignment algorithm as our past studies did, that is to say, the optimal cage is assigned based on the current situation of DDES and the assignment is not changed later. Contrary to this kind of event-driven model, Floor Assignment algorithm is proposed by a timer and event-driven hybrid model. Since the situation of DDES keeps changing, FA is invoked by a preset timer or a cage idling event to assign the optimal destination floor where the idle cages should move to.

5.2.2 Evaluation Items

To determine the optimal floor of idle cages, several evaluation items, i.e., $X \in \{CP, SA, PPA\}$ are proposed as follows. They will be used in the judgment nodes of GNP, and their functions are described later.

Cage Position (CP) To avoid the bunching mode of elevator group systems, which is reportedly linked to a poor performance, the positions of all cages are considered when determining the optimal floor for the idle cages.

Service Area (SA) With the same above reason, service area of each cage is defined for Floor Assignment algorithm since the states of each cage including the moving direction and moving speed are also some important factors.
5.2. Idle Cage Assignment-embedded DDES controller using GNP

Information Management Part

<table>
<thead>
<tr>
<th>Cage ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AT_{sd} )</td>
<td>0.9</td>
<td>0.3</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>( AT_{td} )</td>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>( CHC_{sd} )</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( AT_d )</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>( AET_d )</td>
<td>0.2</td>
<td>0.1</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Cage Position (CP)
- Service Area (SA)
- Predicted Passenger Arrival (PPA)

Fig. 5.1 Outline of the Proposed Method
Predicted Passenger Arrival (PPA)  In an ideal case, the idle cages should be dispatched to the floor where new passengers will arrive in the near future if we can precisely predict the next passenger arrival. PPA is another valuable factor of the proposed evaluation.

5.2.3 Main Algorithm

Fig. 5.2 shows the flowchart of the proposed GNP controller. After the controller is started, GNP will be invoked by some events including the call event and cage idling event or a preset timer. The Cage Assignment algorithm (CA) is invoked and the cage to serve is determined when a hall call with DFGS occurs by a passenger. On the other hand,
the Floor Assignment algorithm (FA) is invoked either when a cage becomes idle or the timer triggers a time-up. Since there might be more than one idle cages in DDES, each idle cage will be checked whether to stay at the current position or move to a specified floor by FA.

Since the Floor Assignment algorithm is the main point of this chapter, we employ the same Cage Assignment algorithm which has been proposed in our past research [37].

When the FA of the GNP controller is invoked, the current position of the idle cage is judged firstly with 3 results, i.e., \{Base, General-Low, General-High\}, which represents where the idle cage is. Then, the processing nodes on the transition route of GNP are activated and the evaluation values of the candidate floors are calculated based on the following Eq. (5.1). This evaluation function to be maximized can be executed with different \( X \) until the node transition transfers to the processing nodes of the floor assignment.

\[
e(f) = \sum_{p \in P} w_p \cdot X(f),
\]

where,

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

\( w_p \) : weights of the floor selection processing node \( p \) (\( w_p \) is optimized during the evolutionary process)

\( X(f) \) : evaluation function of floor \( f \)

The evaluation functions \( X(f) \) of floor \( f \) are shown in Fig. 5.3. In \( CP(f) \), \( f_0 \) represents the current position of the cage running upward. Since the area behind the upward running cage in the figure is very hard to serve as floor assignment floors by this cage in
Fig. 5.3 Evaluation Functions of Floors

the near future, the high priority is given to the floors behind the upward running cage. On the other hand, the low priority is given to the floors ahead the running cage. The function should be reversed when the cage runs downward. In $SA(f)$, $f_0$ represents the current position of the cage running upward, and $f_1$ represents the next stop of the cage. Since the floors around the next stop of the cage are serviced by the cage, its function value is set to the lowest one. In $PPA(f)$, the rate of the passengers emerged at each floor during the past 30 minutes is used to predict the next passenger arrival. The higher the rate of the floor is, the more the floor is considered as the service floor of the idle cage. The sum of the rate is 1.0.

The following Eq. (5.2) is used to determine the service floor of the idle cage when the floor assignment node is activated.

$$f = \arg \max_{f \in F} e(f),$$

(5.2)
where, $F$ : set of the number of floors.

### 5.2.4 Node Functions

The node functions in the proposed method are defined as follows.

**Idle Cage Position Judgment Node**

- Judge the current position of the idle cage ($\{\text{Base, General} - \text{Low, General} - \text{High}\}$).

**Candidate Floor Selection Node**

- Calculate the evaluation values of the candidate floors based on Eq. (5.1).

**Floor Assignment Node**

- Assign the idle cage to serve floor $f$

### 5.2.5 Fitness Function

The same items, which have been proposed in our past research, are employed to evaluate the fitness $F$ of each FA individual. As shown in Eq. (5.3), the first two items, average waiting time and maximum waiting time, are minimized for better performance. The third item is minimized to provide more comfortable riding service, while the last one is minimized to eliminate the loop gene of GNP [37].

$$F = \frac{1}{N} \sum_{n=1}^{N} (t_{n})^2 + w_t \cdot (t_{max})^2 + w_c \cdot (n_{c})^2 + w_l \cdot (l)^2,$$

where,

$N$ : total number of passengers
Table 5.1 Specifications of DDES Simulator.

<table>
<thead>
<tr>
<th>Items</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Floors</td>
<td>16</td>
</tr>
<tr>
<td>Number of Shafts (Cages)</td>
<td>6 (12)</td>
</tr>
<tr>
<td>Floor Distance [m]</td>
<td>4.5</td>
</tr>
<tr>
<td>Max. Velocity [m/s]</td>
<td>2.5</td>
</tr>
<tr>
<td>Max. Acceleration [m/s²]</td>
<td>0.7</td>
</tr>
<tr>
<td>Jerk [m/s³]</td>
<td>0.7</td>
</tr>
<tr>
<td>Cage Capacity [person]</td>
<td>20</td>
</tr>
<tr>
<td>Time for Opening Door [s]</td>
<td>2.0</td>
</tr>
<tr>
<td>Time for Closing Door [s]</td>
<td>2.3</td>
</tr>
<tr>
<td>Time for Riding [s/person]</td>
<td>1.0</td>
</tr>
<tr>
<td>Passenger Density [person/h]</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 5.2 Traffic Flow Ratios.

<table>
<thead>
<tr>
<th>Origin Floor</th>
<th>Dest. Floor</th>
<th>LF</th>
<th>7F</th>
<th>ReGF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lobby Floor(LF)</td>
<td>–</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7th Floor(7F)</td>
<td>25</td>
<td>–</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Rest of General Floors(ReGF)</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$t_n$: waiting time of $n$-th passenger

$t_{max}$: maximum waiting time among $N$ passengers

$n_c$: total number of passengers experiencing one cage service

$l$: number of loops of GNP per one hour evaluation

$w_t, w_c, w_l$: weighting coefficients which are set by trial and error.

5.3 Simulation Results and Discussions

5.3.1 DDES Simulator

The DDES simulator was built based on the specifications shown in Table 5.1. All events are simulated in detail by using 0.1 second time unit. In each time unit, the
5.3. Simulation Results and Discussions

<table>
<thead>
<tr>
<th>Table 5.3 Running Parameters of Floor Assignment GNP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Generation</td>
</tr>
<tr>
<td>Population Size</td>
</tr>
<tr>
<td>—Crossover</td>
</tr>
<tr>
<td>—Mutation</td>
</tr>
<tr>
<td>—Elite</td>
</tr>
<tr>
<td>Node Size</td>
</tr>
<tr>
<td>Crossover Rate</td>
</tr>
<tr>
<td>Mutation Rate</td>
</tr>
<tr>
<td>Evaluation Time [hour]</td>
</tr>
<tr>
<td>( w_f, w_c, w_t )</td>
</tr>
</tbody>
</table>

events of passengers such as arriving at floors, pushing the button of DFGS, getting on and off the cage, are generated according to the O/D table shown in Table 5.2, which represents a typical down peak pattern. In this chapter, a more complicated traffic pattern is simulated by setting the 7th floor, where a high passenger arrival rate is set compared to the other general floors. Table 5.3 shows the running parameters of the proposed method.

5.3.2 Fitness curves

Fig. 5.4 shows the fitness curves of the best individuals during the evolutionary process of GNP with floor assignment individuals. In order to reduce the influence of random noises, we did the experiments using 5 random seeds. Fitness curves of all 5 random seeds are listed in Fig.5.4 as well as the average one. Note that the population of GNP controller was optimized generation by generation and converged to a certain level at the latter generations of the evolutionary process.

5.3.3 Performance comparisons

To confirm the generalization ability of the proposed method, the best individual of each rand seed obtained in the above evolutionary process is tested on the same DDES
simulator for 30 times, 2 simulated hours per each time. To verify the efficiency and effectiveness of the proposed method, the performance comparisons have been done firstly between the proposed method and the method called *Non-FA Method*. In *Non-FA Method*, there is no floor assignment algorithm for the idle cage, that is to say, the cage will stay at the last floor after it serves all registered hall and cage calls.

Moreover, there are six other heuristic methods proposed for some further performance comparisons in this chapter. They are *SA Method*, *SA+CP Method*, *PPA+SA+CP Method*, *Fixed-FA(1F) Method*, *Fixed-FA(7F) Method* and *Fixed-FA(16F) Method*. In *SA Method*, only $SA(f)$ is used in the proposed method to determine the service floor of the idle cage. Similarly, $SA(f) + CP(f)$ and $PPA(f) + SA(f) + CP(f)$ are used as $e(f)$ in Eq. (5.1) in *SA+CP Method* and *PPA+SA+CP Method*, respectively. On the other
Table 5.4 Performance Comparison of Different Methods in Simulations

<table>
<thead>
<tr>
<th>Method</th>
<th>LWR</th>
<th>Imp.</th>
<th>AWT</th>
<th>Imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Method</td>
<td>0.01</td>
<td>77%</td>
<td>10.3</td>
<td>17%</td>
</tr>
<tr>
<td>Non-FA Method</td>
<td>0.04</td>
<td>0%</td>
<td>12.4</td>
<td>0%</td>
</tr>
<tr>
<td>SA Method</td>
<td>0.02</td>
<td>39%</td>
<td>11.9</td>
<td>4%</td>
</tr>
<tr>
<td>SA+CP Method</td>
<td>0.02</td>
<td>39%</td>
<td>12.1</td>
<td>2%</td>
</tr>
<tr>
<td>PPA+SA+CP Method</td>
<td>0.04</td>
<td>7%</td>
<td>13.0</td>
<td>-5%</td>
</tr>
<tr>
<td>Fixed-FA(1F) Method</td>
<td>0.05</td>
<td>-27%</td>
<td>16.9</td>
<td>-37%</td>
</tr>
<tr>
<td>Fixed-FA(7F) Method</td>
<td>0.02</td>
<td>50%</td>
<td>11.6</td>
<td>6%</td>
</tr>
<tr>
<td>Fixed-FA(16F) Method</td>
<td>0.03</td>
<td>11%</td>
<td>15.3</td>
<td>-24%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>AST</th>
<th>Imp.</th>
<th>MWT</th>
<th>Imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Method</td>
<td>33.3</td>
<td>6%</td>
<td>36.8</td>
<td>18%</td>
</tr>
<tr>
<td>Non-FA Method</td>
<td>35.6</td>
<td>0%</td>
<td>45.1</td>
<td>0%</td>
</tr>
<tr>
<td>SA Method</td>
<td>35.6</td>
<td>0%</td>
<td>41.5</td>
<td>8%</td>
</tr>
<tr>
<td>SA+CP Method</td>
<td>35.8</td>
<td>-1%</td>
<td>43.5</td>
<td>4%</td>
</tr>
<tr>
<td>PPA+SA+CP Method</td>
<td>36.7</td>
<td>-3%</td>
<td>44.6</td>
<td>1%</td>
</tr>
<tr>
<td>Fixed-FA(1F) Method</td>
<td>40.6</td>
<td>-14%</td>
<td>48.3</td>
<td>-7%</td>
</tr>
<tr>
<td>Fixed-FA(7F) Method</td>
<td>34.3</td>
<td>4%</td>
<td>35.3</td>
<td>22%</td>
</tr>
<tr>
<td>Fixed-FA(16F) Method</td>
<td>39.1</td>
<td>-10%</td>
<td>47.9</td>
<td>-6%</td>
</tr>
</tbody>
</table>

Note: *Imp.* is defined by \( \frac{\text{Value}_{\text{non-FA}} - \text{Value}_x}{\text{Value}_{\text{non-FA}}} \), \( x \in \{\text{Proposed Method, SA, SA+CP, PPA+SA+CP, Fixed-FA(1F), Fixed-FA(7F), Fixed-FA(16F)}\} \).

Hand, the idle cages are fixedly dispatched to the 1st floor in *Fixed-FA(1F) Method* while 7th floor in *Fixed-FA(7F) Method* and 16th floor in *Fixed-FA(16F) Method*.

The performances of each method are listed in Table 5.4. There are four performance criteria, *LWR*, *AWT*, *AST*, and *MWT*. *LWR* represents the ratio of the long waiting, i.e., the ratio of passengers who wait more than 60s. *AWT* represents the average waiting time of all passengers during the test period. *AST* represents the average system time of all passengers, which is the sum of the average waiting time and the average travelling time. *MWT* represents the maximum waiting time during the test period.

Table 5.4 shows that the proposed method outperforms over the *Non-FA Method* and six other heuristic methods except *MWT* by the *Fixed-FA(7F) Method* on all four performance criteria. The *Imp.* columns of each performance criteria show the percentage
improvement of each method comparing with the *Non-FA Method*. The worst performances of the *Fixed-FA*(1F) *Method* on all four criteria suggest that inappropriate floor assignment for the idle cage, in this case the idle cage is dispatched to the 1st floor regardless of the down-peak traffic pattern, will deteriorate the performance to some extent.

In addition, as mentioned earlier, the idle cage assignment algorithm embedded controller is proposed especially for a light traffic mode where some idle cages exist. That is to say, the proposed method would not contribute to the system performances when it is employed in a heavy traffic mode, because there are almost no idle cages in such a mode. Furthermore, the idle cage assignment algorithm even deteriorate the performance a bit when there are only few idle cages in a moderate traffic mode, because the cage movement
5.4 Conclusions  

In this chapter, we proposed an idle cage assignment algorithm-embedded DDES controller for the light traffic mode where some idle cages exist. Three evaluation items are proposed to determine the service floor for the idle cage, and it is selected with optimized weights during the node transition of the proposed GNP. Fitness curves show that the evolutionary process has been done generation by generation. The best individuals are firstly applied to a light traffic mode (200 persons/h) and compared with the Non-FA Method and six other heuristic methods. The efficiency and effectiveness of the proposed method have been verified by the performance comparisons. Furthermore, the proposed method has been applied to various traffic modes from light to heavy to clarify its applicable conditions which has been supposed earlier in this chapter.
Chapter 6

Conclusions

In this research, some studies on Genetic Network Programming-based controller of elevator group systems were done in terms of multiple objectives. For faster training speed and better performances of elevator group systems, Reinforcement Learning (RL) was introduced into the GNP controller of Single-Deck Elevator Systems (SDES) and Double-Deck Elevator Systems (DDES) considering their specific features respectively. For adapting the traffic flow in buildings, a function localized framework of GNP was proposed for DDES. Finally, in order to improve the overall performances of DDES, the idle cage was proposed to be dispatched to some floor in the light traffic mode.

Motivated by the past research of applying GNP to EGSCS and applications of GNP with RL to virtual worlds, an elevator group controller using Genetic Network Programming (GNP) with Reinforcement Learning (RL) was proposed in chapter 2. Experiments have been done in two steps to verify the efficiency of the proposed method. Firstly the evolutionary and learning process of the proposed method has been run in a certain traffic mode. Performance results showed that the proposed method outperformed other three methods in all three traffic patterns except up peak. After that, the generalization ability has been also checked under various traffic densities. Moreover, another experiment called GNP-RL with weight tuning by tuning the importance weight of the macro-processing nodes has been also done as well as some studies on its effectiveness.

After the applicability and effectiveness of GNP with RL in elevator group systems have been clarified in chapter 2, this approach has been redeveloped for Double-Deck
Elevator System (DDES) in chapter 3 with the specific features of DDES considered. To verify the efficiency and effectiveness of the proposed method, experiments have been done on a detailed DDES simulator in three typical traffic patterns. The function of RL has been verified by comparing the fitness curves between the method with RL and without RL. In addition, the performances of the proposed method in the generalization tests showed the improvement over other methods including the THV Method which has been originally proposed for single-deck elevator group systems.

In chapter 4, a traffic-adaptive DDES controller has been proposed considering the varying traffic flow in real application situation since all previous studies proposed to optimize the GNP controller firstly in three typical traffic patterns and then applied them to corresponding traffic pattern. In the proposed method, the cage selection part has been designed to be functionally localized by the evolutionary process to make a flexible cage assignment for the varying traffic flows. Analyses on the results of function localization in cage selection part have been done according to simulation output data as well as the switching results of the switching method.

So far, most of studies including those in this research contribute to provide some optimal solutions for the heavy traffic mode of elevator group systems. The overall performances of elevator group systems, however, are generally evaluated not only in heavy traffic mode but also in light traffic mode. In the last chapter, chapter 5, an idle cage assignment algorithm-embedded DDES controller has been proposed for the light traffic mode where some idle cages exist. Simulation results showed that the proposed method is efficient in light traffic mode and becoming trivial when the traffic density reaches a higher level, as what has been supposed in the proposal.
References


Appendix A

Genetic Network Programming (GNP)

A large number of studies on the evolutionary optimization techniques have been made. Genetic Algorithm (GA) [24], Genetic Programming (GP) [25, 26] and evolutionary Programming (EP) [42, 43] are the typical methods of the evolutionary algorithm. GA evolves strings and it is mainly applied to optimization problems. GA can find suboptimal solutions of the problems quickly, so it has been widely studied and applied to many real problems [44]. 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. Therefore, it is important how to express solutions for the optimization problems because the form of solutions limits the problems to be solved.

Basically, GNP is an extension of GP in terms of gene structures [27, 28, 45–49]. The original idea is based on the more general representation ability of graphs than that of trees. The aim of developing GNP is to deal with dynamic environments efficiently by using the higher expression ability of graph structure than that of trees, and the inherently equipped functions in it.

A.1 Basic Structure of GNP

The basic structure of GNP is shown in Fig.A.1. As shown in Fig.A.1, the directed graph structure is used to represent individuals. GNP is composed of plural nodes which are roughly classified into two kinds of node: Judgment node and Processing node.
Fig. A.1 Basic structure of GNP
Judgment nodes correspond nearly to elementary functions of GP and processing nodes correspond almost to terminal symbols of GP. Judgment nodes are the set of $J_1, J_2, \cdots, J_m$, which work as some kinds of judging functions. On the other hand, Processing nodes are denoted by the set of $P_1, P_2, \cdots, P_n$, which work as some kinds of action/processing functions. The practical roles of these nodes are predefined and stored in the library by supervisors.

The connection is branched off by the judgment results, which are predefined by judgment functions in Judgment nodes. Accordingly, if there are a lot of judgment results, the number of branches increase, the network structure become complicated. And Processing nodes have just one branch in order to carry out the next judgment (processing). Additional specific nodes, Start node $S$, is involved in GNP. Start node indicates the start point of GNP, which corresponds to GP $\mathbb{L}$s root node.

GP $\mathbb{L}$s elementary functions and terminal symbols are repeatedly used in a tree structure. In the same way, there are some $J_1, J_2, P_1, P_2$, and so on in GNP as shown in Fig.A.1. These Judgment nodes and Processing nodes are the essential elements of GNP. The number of these nodes may be determined as a result of evolution like GP. Actually, GNP can use this strategy, in other words, GNP can adopt evolving the genotypes with variable number of nodes, but in this research, GNP evolves only the networks with the predefined number of nodes. It would be better to say that GNP here evolves the genotypes with fixed number of nodes. We set the number of each node in GNP equal to each other, e.g., $J_1 \times 3, J_2 \times 3, \cdots, P_1 \times 3, P_2 \times 3$, and so on.

Once GNP is bootstrapped, the execution starts from the Start node, then the next node to be executed is determined according to the connection from the current activated node. If the activated node is Judgment node, the next node is determined by the judgment results. When Processing node is executed, the next node is uniquely determined by the single connection from Processing nodes.
As mentioned above, GNP having a basic structure deals with the unit of each program as individual and makes up the population (group) gathered by individuals. Basically, GNP finds adaptive solutions by carrying out the evolution in the population.

The individuals moving to the next generation in which the genetic operation is carried out are selected with the fitness. After all, the result to be executed by the program could be obtained, and the fitness of solution is calculated for each individual.

### A.2 Genotype and Phenotype of GNP

It is necessary to encode genetic information properly of each node in order to express GNP in a computer. As shown in Fig.A.2, it is the translation from the phenotype as the network structure to the genotype carrying out programming for calculation on a computer. The phenotype and genotype of GNP is shown in Fig.A.2. The genetic information expressing kinds and connections at each node are written in the genotype, then the its set is defined as a GNP program.

The genotype expression of GNP node is shown in Fig.A.2. This describes the gene of node \( i \), then the set of these genes represents the genotype of GNP individuals. All variables in these genes are described by integer. \( NTi \) describes the node type, \( NTi = 0 \) when the node \( i \) is start node. \( IDi \) is an identification number, e.g., \( NTi = 1 \) and \( IDi = 1 \) mean node \( i \) is \( J_1 \). \( C_{i1} C_{i2} \cdots \) denote the nodes which are connected from node \( i \) firstly, secondly, \( \cdots \), and so on depending on the arguments of node \( i \). The total number of connection genes depends on the parity of the node \( \Delta s \) function. \( d_i \) and \( d_{ij} \) are the delay time. They are the time required to execute the processing of node \( i \) and delay time from node \( i \) to node \( C_{ij} \), respectively. GNP can become materialized more realistically by setting these delays.

The only connection genes is basically different in all individuals of the population carrying out an evolution. In other hand, the number of all nodes are the same and
A.2. Genotype and Phenotype of GNP

![Diagram of GNP node with genotype and phenotype expressions]

<table>
<thead>
<tr>
<th>Node</th>
<th>NT</th>
<th>ID</th>
<th>d_i</th>
<th>C_i1</th>
<th>d_i1</th>
<th>C_i2</th>
<th>d_i2</th>
<th>\ldots</th>
<th>C_in</th>
<th>d_in</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. A.2 The genotype and phenotype expression of GNP node.
nodes having the same identification number have the same node functions among all individuals (there are cases where different node functions exist by the genetic operation such as mutation or crossover.). As mentioned above, nodes are transferred by the genetic information.

### A.3 Genetic Operation of GNP

Although GNP is a technique discovering optimum solutions by evolving a program, we use a technique based on the genetic operation of GA. Selection, mutation, and crossover established by the genetic operation of GNP are reviewed.

#### A.3.1 Selection

The individuals carrying out crossover or mutation and individuals preserved to the next generation are selected according to rules based on the fitness of each individual. The following selections shown in Fig.A.3 are established in GNP.

- **Roulette Selection**

  As the roulette selection of GNP, the selection is carried out by the probability in proportion to the relative value of the fitness of the individual. The roulette selection has
not been used in conventional research because it could not be applied to the case where lower fitness is dominant [49].

- **Tournament Selection**
  Comparing with $N$ individuals selected from the population randomly, the individual having the highest fitness is selected among them. $N$ is the tournament size and $N = 2$ is used generally. The tournament selection is mainly used in conventional GNP research, because the tournament selection is available in the case where lower fitness is dominant.

- **Elite Selection**
  Comparing with the fitness of all individuals of the population, the elite selection moves $M$ individuals having higher fitness to the next generation. $M$ is the number of elite individual, and the solution converges quickly if $M$ is high.

### A.3.2 Crossover

The crossover of GNP based on GA is the genetic operation generating two new offspring by exchanging the genetic information among the genotype of two parents. Exchanging the genetic information in the genotype is the same to the exchange of the subnetworks of GNP in phenotype. The following crossovers are established in GNP.

- **One Point Crossover**
  Selecting one node as the crossover point randomly, the whole genetic information of its node is exchanged. The example of one point crossover is shown in Fig.A.4. The performance of the generated offspring individual is influenced by the position of the crossover point.

- **Several Points Crossover**
  Plural nodes (generally two) are selected as the crossover point randomly, the whole genetic information of their nodes is exchanged. The example of several points crossover
is shown in Fig.A.5. Exchanging more small sub networks is available by dividing the network into more blocks.

- **Uniform Crossover**

Which nodes are better is decided by the predefined crossover probability $P_c$ for each node of the parent individual, the whole genetic information corresponding to the crossover node is exchanged between the parents. The example of uniform crossover is shown in Fig.A.6.

**Fig. A.4 One Point Crossover examples of GNP**
A.3.3 Mutation

The mutation is carried out by changing genetic information randomly in the mutation of GNP based on GA. The kinds of the mutation are classified according to the changing information as follows:

- Mutation of connections between nodes

The connection between nodes is modified. The mutation refers to the change of genetic information $C_{ij}$ of each node in GNP by the predefined probability $P_{mc}$, and decides the
connection for the mutation. Therefore the genetic information \( C_{ij} \) is modified in the range of the node number randomly. The example of the mutation is shown in Fig.A.7.

In the case of just carrying out this mutation, the kinds of nodes having the same node number are not modified because \( NT_i \) and \( FID_i \) are not modified.

- **Mutation of nodes**

  The kinds of node are changed. The mutation refers to the change of genetic information \( NT_i \) and \( FID_i \) of each node in GNP by the predefined probability \( P_{mn} \), and decides the node and the genetic information for the mutation. Therefore, the genetic information on the \( NT_i \) and \( FID_i \) is modified randomly. If the number of the connections are
increased by modifying the node function (e.g., modifying the Processing node to the Judgment node), added connections are defined randomly. On the other hand, if the number of the connections are decreased, they are deleted. The example of the mutation is shown in Fig. A.8.
Appendix B

Genetic Network Programming with RL

As described in App. A, 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.

B.1 Structure of GNP with RL

Figure B.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. B.1, $K_i$ represents the node type, which is the same as original GNP. $W_i$ is proposed in this research as the importance weight\(^1\) of each node and it can be optimized through the mutation process described later. $ID_{ip}(1 \leq p \leq m_i)$ shows the identification number of the node function. If $m_i$ of all nodes is set at 2, GNP can have the node function $ID_{i1}$ and $ID_{i2}$ in the node. $Q_{ip}$ means Q value which is assigned to each state and action pair. In this method, “State”

\(^1\)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.
means a current node, and “Action” means a selection of node function $ID_{ip}$. Note that the actions of agents and the actions in the RL of GNP are not the same. $d_{ip}$ is the time delay spent on judgment or processing. $C_{ip}^A$, $C_{ip}^B$, ... show the node number of the next node $j$, i.e., $j \in \{C_{ip}^A, C_{ip}^B, ...\}$. $d_{ip}^A, d_{ip}^B, ...$ mean the time delays spent for the transition from the sub node with $ID_{ip}$ to node $C_{ip}^A$.

### B.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.B.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.

#### B.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 research, 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 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 research, 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 $\pm 0.1$ with probability 0.5.

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

B.2.2 Learning process

As mentioned above, a state means a current node and an action means the selection of a function. Fig.B.3 shows states, actions and an example of node transition, where there are $m_i$ sub-nodes in a macro-node. Learning process is explained as follows based on that example using processing nodes.
(1) At time $t$, GNP refers to $Q_{i1}, Q_{i2}, \ldots, Q_{im}$ \textsuperscript{2} and selects one of them based on $\varepsilon$-greedy policy. We suppose that GNP selects $Q_{ip}$ and the corresponding function $ID_{ip}$.

\textsuperscript{2}The initial $Q$ values are usually assigned to zero.
(2) Then GNP executes the function $ID_{ip}$, gets the reward $r_t$ and the next node $j$ becomes $C^A_{ip}$.

(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 $C^A_{ip}$, $C^B_{ip}$, ... according to the judgment result.
Appendix C

Elevator Group Supervisory Control System (EGSCS)

C.1 Outline of 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.

C.2 Cage Assignment Policies in EGSCS

There are two kinds of cage assignment policies. One is called immediate policy, which makes an immediate cage assignment based on the current system information and does not change it even though another better solution of cage assignment is available later. The shortage of the suboptimal solution in this policy is compensated by giving a guidance
message which can palliate the tension of waiting passengers and have them move in front of the assigned elevator in advance. The other one is called reassignment policy, which makes the cage assignment in an appropriate timing or reassigns the past cage assignment. It tends to find the optimal solution using the latest information of the elevator group system, but much more computational costs are needed. Most of studies have been done by employing the former policy while few such as [50] has been reported by employing the latter one.
C.3 Traffic Patterns in EGSCS

Different buildings have different passenger traffic patterns. For example, the traffic pattern in commercial building is totally different from the pattern in residential building. The traffic pattern in commercial building is generally studied in most of research. There are three traffic patterns defined in a typical office building, i.e., “Up Peak Time”, “Down Peak Time” and “Regular Time” (also called off-peak time).

In “Up Peak Time”, almost all of passengers arrive at the lobby floor and travel to upper floors which usually occurs in the morning. By contrast with this, almost all of passengers wait at upper floors heading for the lobby floor in “Down Peak Time” which usually occurs in the evening. In “Regular Time”, passengers move between the lobby floor and upper floors in two directions without a dominated trend. There actually exists another traffic pattern called “Lunch Time”, in which the traffic flow between lobby floor and upper floors is drastically increased for having lunch. This traffic pattern can be approximated by 45% “Up Peak Time”, 45% “Down Peak Time” and 10% “Regular Time”.

C.4 Performance Criteria of EGSCS

There are several performance criteria in EGSCS. Some are considered when designing an EGSCS such as Handling Capacity and Round Trip Time, some are employed to evaluate the developed controller of EGSCS such as Average Waiting Time, Long Waiting Ratio, and so on. They are listed as following.

- **Handling Capacity (HC)**: The percentage of the building population served by elevator group systems within five minutes.

- **Round Trip Time (RTT)**: The span between the time a cage closes its door and leaves the lobby floor and the time the cage returns to the lobby floor and opens its door after finishing all registered hall/car calls.
• **Average Waiting Time (AWT)**: The waiting time averaged over all passengers during a certain period.

• **Average Traveling Time (ATT)**: The averaged time which is taken to deliver the passenger on board from his/her origin floor to the destination floor.

• **Average System Time (AST)**: The sum of AWT and ATT.

• **Maximum Waiting Time (MWT)**: The maximum time which is taken by the passenger to wait for the cage service.

• **Long Waiting Ratio (LWR)**: The ratio of the passengers who wait for the cage service over 60 seconds during a certain period.

• **Energy Consumption (EC)**: The amount of energy consumed during a certain period, which is usually measured by the movement distance of cages.

It is impossible to optimize all the above performance criteria at the same time due to the inherently contradictory nature of each one. That is to say, when optimizing one criterion, the other criteria may suffer. For example, the attempt to decrease the average waiting time would always result in an increase of energy consumption. Most of approaches are proposed to optimize EGSCS based on multiple weighted performance criteria. [51] is one of them where the best weights are optimized by neural networks.

In this research, AWT, AST, LWR and MWT are employed as the optimization targets. Particularly, AWT is introduced into the fitness function with squared waiting time instead of the original one for fairer service. For instance, two sets of waiting time (6 and 8 seconds, 4 and 10 seconds) with the same average (7 seconds) have the same priority while their squared averaged value are 50 and 58 respectively with a higher priority (fairer) for the former case.
Appendix D

Double-Deck Elevator Systems (DDES)

D.1 Outline of DDES

The double-deck elevator system (DDES) was developed in 1930’s to improve the transportation capacity of elevator group systems in high-rise buildings without adding more elevators in place of single-deck elevator systems (SDES). Fig. D.1 shows the outline of DDES. The main difference of the hardware configuration in DDES is that there are two cages connected vertically in each shaft, which improves the transportation capacity significantly in an up-peak traffic pattern, especially in a pure up-peak one.

On the other hand, the specific features due to the hardware configuration make the control system more complex, deteriorating the performance of DDES drastically in other traffic patterns like regular and down-peak time. This is the reason why the DDES has not been widely 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 [52]. The recent invention of the destination floor guidance system (DFGS), where a destination operation panel is installed in each floor instead of up and down call buttons, provided an opportunity to develop some more efficient control algorithms for DDES. A detail-simulated DDES with DFGS is employed as the testbed in this research.
D.2 Running modes of DDES

There are three kinds of running modes in DDES. Generally speaking, these modes can be switched if the elevator group controller supports to do so.

**Double running mode**  In this mode, the upper/lower cage serves only odd/even floors, respectively. With the cut of about half of the total stops, the transportation capability of DDES can be doubled in a PURE up-peak traffic pattern. On the other hand, the movement of the passengers between odd and even floors would be restricted in this running mode.
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. All proposals in this research is based on this running mode.

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, DDES runs like a SDES in this mode.

D.3 Specific features of DDES

Compared with SDES, there are several specific features in DDES, where two decks in each shaft are connected. These features shown in Fig. D.2, of course, are important and necessary to be considered when developing a control algorithm for DDES.

One cage service  The elevator stops at a floor with only one cage serving passengers, which means the other cage has to stop without any passengers boarding or exiting, even when no need to open its door. This case should be avoided as much as possible, since it will make those passengers without service in the cage uncomfortable.

Coincident service  Both cages serve passengers at two adjacent floors simultaneously with one stop. As we can imagine, this case is given the highest priority when making cage assignment, since it contributes directly to the improvements of the transportation efficiency of DDES.

Separate riding for identical destination  Passengers for an identical destination ride on both cages, which means the elevator has to stop twice at the same floor. This case should be also avoided as much as possible, since it will deteriorate the system’s transportation efficiency and may cause much more one cage service.

Unbalanced load  The number of passengers or registered calls between the upper and lower cages is unbalanced. Decrease of the system’s transportation capability can be linked to this case.
D.4 Cage assignment policy

As introduced in App. C.2, there are two main cage assignment policies in the elevator group control system, immediate assignment policy and reassignment policy. Even though the latter one could also be employed in DDES, with the DFGS introduced into DDES in this research, we adopt the former one since it is usually required to make the cage assignment decision and give the waiting passengers a guidance message immediately after the destination floor number is entered by the coming passengers.

Fig. D.2 Specific Behaviors in Double-Deck Elevator Systems.
D.5 Destination Floor Guidance System

As one of the approaches to improve the performances of elevator group systems, Destination Floor Guidance System (DFGS) [53–55] has been proposed and widely installed in many real elevator systems recently.

As shown in Fig. D.3, a destination input panel is generally installed at the entrance of the DDES. Passengers are required to enter their destination floor number before they ride on the escalator to the right lobby. The assigned cage number is to be shown on the panel guiding the passengers to board the corresponding cage. Since the destinations of most passengers if not all\(^1\) are known, the controller can make a more efficient cage assignment decision by group those passenger with the same destination.

\(^1\)For a group of passengers who have the same destination, usually only the first person will enter the destination when there is no requirement for each passenger to input his/her destination.
Appendix E

Framework of Switching Method

To adapt to the varying traffic flows, there exits another method switching the GNP controllers which are evolved and optimized for three different traffic patterns (up-peak time, regular time and down-peak time) in advance. App. Fig. E.1 shows the outline of this method. The parameters of the GNP switch controller for classifying the traffic flow are the same as the ones of the proposed method in chapter 4, and they are optimized by using real-coded GA with non-uniform mutation for a precise tuning.

Fig. E.1 Outline of the Switching Method
The following fitness function, which is similar to the one used in the proposed method shown as Eq. (4.2) in section 4.3 of chapter 4, was employed to evaluate the individuals in the switching method. Note that the last term, $w_t \cdot (N_t)^2$ is not included in Eq. (E.1) since there does not exist a loop when tuning the parameters during the evolutionary process.

$$f = \frac{1}{N} \sum_{n=1}^{N} (t_n)^2 + w_t \cdot (t_{max})^2 + w_c \cdot (N_c)^2$$  \hspace{1cm} (E.1)

We used the same traffic data as the proposed method for training the switching method shown as Fig. 4.1 in section 4.2 of chapter 4. The fitness curves of 5 random seeds are shown in app. Fig. E.2 as well as their average. Note that the average fitness starts
at 1550 and ends at around 1500 with some fluctuations. The reason why the fitness function does not improve so much is that three kinds of GNPs already optimized to up-peak time, regular time and down-peak time are just switched.

### E.2 Switching Results of Switching Method

App. Fig. E.3 shows the switching results of the switching method obtained from the actions of the GNP switch controller on a work day traffic data. Note that the switches during 9:30-11:30 and 13:30-17:30 are made by GNP switch controller though there is no large change in the traffic flows, which might potentially deteriorate the performances of the switching method.
Acknowledgments

First I would like to thank Prof. Kotaro Hirasawa for providing me the opportunity to join his team, giving me a kindly guidance on my research and teaching us how to improve our abilities adapting to society life. During the past five years, his enthusiasm, concentration and persistence on research, his kindness on supervision impressed me deeply. I am glad to have been his student and thank him once again for what he taught us, what he showed us so far. And then, I would like to thank Prof. Osamu Yoshie and Prof. Toshihiro Nishimura for their valuable advices on the revision of this doctor thesis.

I am also grateful to professor Jinglu Hu for giving me some constructive suggestions and many helps both in my research and living life.

Many thanks are due to Dr. Eguchi who gave me a great help on my research. And also thanks to Dr. Mabu, Dr. Eto, Dr. Shimada, Ms. Yu Lu, Ms. Johanna and the rest of members of our lab for their helpful advices and enjoyable communications.
Research Achievements

Journal Paper


**International Conference Paper**


Domestic Conference Paper

(1) 坊金，Yu Lu，間津真吾，平澤宏太郎，マルコンシャンドル，“Genetic Network Programming によるダブルデッキエレベータ群管理システム”，計測自動制御学会，システム・情報部門学術講演会，pp. 131-135, 2006/11.