In this paper, a reinforcement learning method is proposed that optimizes passenger service in elevator group systems. Task-oriented reinforcement learning using multiple agents is applied in the control system in allocating immediate landing calls to the elevators and operating them intelligently in attaining better service in this stochastic dynamic domain. The proposed system shows better adaptive performance in different traffic profiles with faster convergence compared to the other learning elevator group control system.

**Keywords:** elevator group control, task-oriented reinforcement learning, multiagent systems

1. Introduction

Most of the real world control problems have complex objectives, and there is rarely a known optimal policy. In such cases, trainable systems such as reinforcement learning (RL) agents can successively improve their control strategies through interactions with the environment. Elevator group control (EGC) system is such a difficult domain that poses combination of high dimensional state space and complex dynamics due to stochastic traffic patterns, and is to be managed in order to transport the passengers efficiently. An EGC system has to control elevators in the system considering the current situation as well as many uncertain factors such as destinations of the incoming passengers, arrivals of future passengers, movement of other elevators etc.

Significant progress have been made in designing control system for elevator groups using heuristic methods showing more desirable results than conventional statistical systems. Fuzzy logics are widely used in the recently proposed EGC systems, either to classify the traffic pattern or to assign the hall calls\(^{(1)(2)}\), but none of them provides the optimal solution\(^{(3)}\). Trainable systems seem very promising for this complex problem area, Crites et al demonstrated a collective multiagent reinforcement learning algorithm for a typical simulated elevator group system. The system outperformed other EGC systems for pure as well as mixed down-peak passenger traffics\(^{(4)}\), but it takes extremely long time to converge that makes the system difficult to implement in real EGC systems.

In this paper, we propose a faster and more efficient method for EGC system using multiagent task-oriented reinforcement learning (TORL). TORL, whose first version has been proposed by the authors\(^{(5)-(7)}\), is a framework for solving large-scale problems by reinforcement learning. It decomposes the given task into subtasks, which reduces the state space size. The subtasks are defined from the viewpoint of task, which enables sharing of a single policy among agents and farther reduces the state space size. The EGC problem is redefined within the TORL framework.

The simulation results show significantly improved performance in controlling this very large-scale stochastic dynamic system compared to the other adaptive learning method.

Next section introduces the elevator group control domain and related research. Section 3 describes the basic notion of the conventional and the task-oriented approaches of reinforcement learning, and the implementation for elevator group control. Section 4 contains simulation results, Section 5 contains discussions on the simulation results, and finally conclusions are drawn in section 6.

2. The Elevator Testbed

Real time control of an elevator group for a twelve-storeyed residential building with two elevator cars serves as our test bed. The objective considered in this system is to minimize the average waiting time of incoming passengers. A schematic diagram of the elevator system is presented in Fig. 1. The elevator cars are represented as filled boxes in the diagram. ‘+’ represents a ‘hall call’ or someone waiting to enter a car. ‘−’ represents a ‘car call’ or someone wanting to leave the car. The left side of a shaft represents upward moving cars and calls, whereas the right side of a shaft represents downward moving cars and calls.

Simplest way to reduce the passengers’ waiting time is to add more elevator cars, but it requires operating and installation cost of additional elevators. Minimizing
the waiting time of incoming passengers using only two elevator cars in a high storeyed building is very difficult task, which is the main objective of this test bed.

2.1 Complexity in Elevator System The elevator system poses significant difficulties as it operates in a high-dimensional continuous state space and in real time as a discrete event dynamic system. Its state is not fully observable, and it is non-stationary due to changes in passenger arrival rates. The elevator system is driven by passenger arrival which is stochastic and varies during the course of the day with different peak and off peak periods. In typical residential building, the morning rush brings a peak level of down traffic while a peak in up traffic occurs during evening, and moderate or light profile in up or down traffic exists in other times.

Besides the standard service provided by the elevator during operated by the inside passengers (passengers in an elevator car), it is necessary to make some critical decisions such as which elevator should respond when a hall call button is pushed by an incoming passenger (a passenger who wants to get in an elevator car). Moreover, when an elevator car is idle, it is desirable that it waits at the best floor to get maximum passengers.

2.2 Related Strategies A number of researchers have proposed different heuristic approaches in controlling the elevator groups whereas a little work has been done in adaptive and learning approaches particularly using reinforcement learning. Levy et al used dynamic programming (DP) to minimize the expected time needed for completion of the current busy period. To keep the system manageable, the number of buttons that can be on simultaneously is restricted and unlimited capacity of elevator cars is considered. Major difference with RL is that it must be performed off-line since it uses a model of the transition probabilities of the system and performs sweeps of the state spaces.

Crites et al proposed a multiagent reinforcement learning algorithm for EGC using global reinforcements for the team of agents in terms of waiting time of incoming passengers. In this multiagent system, each elevator car is operated by an RL agent to decide the next action at the restricted decision points, either to stop at the next floor or continue past the next floor. It provides a collective form of RL (CRL) algorithm for the team as a whole and surpasses the best of other heuristic elevator control algorithms in handling pure as well as mixed down peak traffic. The major drawbacks of this collective RL algorithm are: it requires extremely long time (60,000 hours of simulation) to converge as the state space is extremely large and there are possibility of instability, i.e. all elevators may respond to a single hall call at low traffic hours rushing to the calling floor or all of them may not stop at the called floor mistakenly presuming other elevator is going there.

3. Implementation of Elevator Group Control System

3.1 Task-Oriented Reinforcement Learning

Reinforcement learning is a process of trial-and-error whereby an agent seeks to find the combination of actions that maximizes the rewards as its performance feedback. One of the most commonly used reinforcement learning methods is Q-learning. This algorithm does not need a model of the environment and directly computes the approximate function of optimal action-value independent of the policy followed. The updating rule of Q-learning is as follows:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]
\]

where, \(\alpha\) is the learning rate, \(\gamma\) is the discount factor, \(r_t\) is the achieved reward at time \(t\), and \(Q(s_t, a_t)\) is the value of action \(a_t\) at state \(s_t\).

The agent uses all perceivable information of the environment to constitute the state and tries to choose better action maintaining a balance between exploration and exploitation according to a certain policy. The convergence of the Q-learning is proven under the assumption that each state-action pair is visited infinitely often. Unfortunately, most of the complex problems have high dimensional state space, and it is hardly possible to visit all states in a short interval to reach convergence and almost impossible to converge in dynamic systems.

The task-oriented approach of RL reduces the complexity of the problem by learning from the viewpoint of each logical subtask considering only its related information. For each subtask, a separate lookup table is used to store Q values, where the corresponding agent uses only the specified information of the environment to understand the present condition of the task in term of ‘task-state’ and chooses an action. The goal of each subtask may be different apparently, but it helps attaining the global goal of the system. This method provides one lookup table for each subtask, therefore, the same agent may deal with all lookup tables or a separate
agent can be used for each subtask depending on the involvement of the agent. The main objective of this method is to simplify the learning process considering only necessary information related to the corresponding task by reducing the state-space size, hence it requires less memory, and faster convergence can be achieved.

Figure 2 shows a comparative notion of task-oriented and conventional RL: in the conventional RL system, the agent learns to update the policy by choosing an action considering the state of the environment without specific idea on task; in the task-oriented RL, the agent learns to update the policy by choosing an action considering the state of task from the viewpoint of the task. In the task-oriented system, policies belong to a task instead of the agent, which provides an opportunity of agents’ cooperation by updating common policy instead of the agent, which provides an opportunity of task. In the task-oriented system, policies belong to a task instead of the agent, which provides an opportunity of agents’ cooperation by updating common policy instead of the agent, which provides an opportunity of task.

3.2 Control Strategy
To apply the RL algorithm, the system is modeled as a discrete event system representing its dynamics in terms of small time units. For the task-oriented approach, the whole task is decomposed into smaller subtasks. In this EGC system, there are three kinds of tasks which should be performed for its successful operation. These tasks are parts of the function of the elevator group. The first task is operating the elevator according to some priority rules considering commands of inside passengers and calling passengers. An elevator must stop at the next floor if the hall call button is on or if the inside passenger wants to get out there. The second task is to select the appropriate elevator when any hall call button is pressed. The third task is to find the best floor to park the elevator car at its idle time for further reduction of waiting time of forthcoming passengers. The elevator should stay at the floor where the possibility of getting a passenger is higher.

The elevator group control can be done by a single controller, however the control has a number of different aspects. So it should be easier to design a controller as a collection of agents each of which takes care of one or a few control aspects. To solve the above three tasks we introduce two kinds of task-oriented agents: elevator-agents and calling-agents.

Each elevator-agent operates an elevator-car considering the commands of inside-passengers and the calls from calling-agents, according to some priority rules and restrictions without learning (the first task).

There are 22 hall call buttons (two for each floor, except the first and twelfth floors which have only one button). For each of these hall call buttons, a calling agent is assigned and learns independently to choose the most suitable elevator after being called by a passenger (the second task). It is not necessary to implement a controller for each of 22 call buttons. A single controller can take care of all of the virtual agents. In this call allocation task, the state is constituted by the following information: the relative position of an elevator with respect to the call direction (beyond the current floor or not: 2 values), elevator’s moving status (same or opposite to the call direction: 2 values), the distance of an elevator from the calling floor (11 values) and the destination of the elevator (the current floor or other floor: 2 values). Thus there are \(2 \times 2 \times 11 \times 2 = 88\) possible state-values for an elevator. So for the group of two elevators only \(88^2 = 7744\) states are considered in this call allocation task. The state information is not specific for a particular calling agent at a floor as the relative information of an elevators from the calling floor with respect to the call direction is considered (e.g., the state information for an up call at the 3rd floor with an up-going elevator at the 2nd floor is the same as that for a down call at the 8th floor with a down-going elevator at the 9th floor; in both cases distances are one-floor, elevator directions are the same as call direction and elevator positions are in behind the calling floor). After each action, a calling agent receives a reinforcement \(R_{\text{call}}\) according to the following eq. (2)

\[
R_{\text{call}} = -a \times T_{\text{wait}} \tag{2}
\]

where \(T_{\text{wait}}\) is the waiting time of the first passenger who pushed the hall call button and \(a\) is a positive constant. The negative sign means negative reinforcement (punishment) proportional to the waiting time. The waiting times of other passengers in the queue are ignored since they are unavailable to the system.

The elevator agent carries out a separate task which demands learning: it learns to find the best floor where the elevator should wait for the future passenger when it is idle (the third task). The best floor means that the floor has the highest probability of having a future

\[\text{Figure 2. Comparative notion of (a) conventional reinforcement learning system, and (b) task-oriented reinforcement learning system} \]
incoming passenger. This is done only when the elevator has no signal from any inside or incoming passenger for a certain interval of its idle time. In this learning, the idle elevator-agent has to choose the suitable floor as its action where it may get a passenger earlier. For this task only the information of the current location (12 floors: 12 values) and status (moving-up, moving-down or idle: 3 values) of the other elevator, and the direction of relative traffic flow (peak-up, peak-down or normal as described below: 3 values) are enough to describe the state to the elevator-agent, which makes the state space size as small as 12 × 3 × 3 = 108 states.

The relative traffic flow \( T \) is estimated by eq. (3),

\[
T = \frac{T_{UP} - T_{DOWN}}{T_{UP} + T_{DOWN}} \times 100,
\]

where \( T_{UP} \) and \( T_{DOWN} \) are the numbers of recent up-going and down-going passengers, respectively. For example, \( T_{UP} \) is incremented when the up-button is pushed at the lobby

\[
T_{UP} \leftarrow T_{UP} + 1.
\]

Since the traffic profile changes with time, the exponential forgetting is introduced

\[
T_{UP} \leftarrow \lambda T_{UP},
\]

where \( \lambda \) is the forgetting factor (0 < \( \lambda \) < 1). The elevator-agent estimates the relative traffic flow direction as: ‘Peak-up’ when \( T > 20\% \), ‘Peak-down’ when \( T < -20\% \) and ‘Normal’ otherwise.

After each action, when the elevator is called, it receives a reinforcement \( R_{elev} \) according to the following eq. (6)

\[
R_{elev} = -b \times d_{floors}
\]

where \( d_{floors} \) is the distance of calling floor from the floor of idled elevator and \( b \) is a positive constant. If the parked elevator is called from the same floor then it receive a reinforcement equal to 0 and any call from the floor other than the parking floor gives negative reinforcement (punishment).

Figure 3 shows a brief explanation of selecting actions by the elevator-agent. The elevator-agent chooses an action (going to 4th-floor) after 200 sec of idled time, the elevator goes to that floor and stays there for 200 sec and finds no call from any floor (i.e. reward 0). So the agent updates the policy and again chooses the action (going to 4th-floor). After staying there for 90 sec it receives a call from 10th-floor, which gives a punishment proportional to \( d_{floors} \), and it updates the policy again.

In this task-oriented implementation, all calling-agents have similar tasks in different floors and the state contains the relative information of the calls and positions of elevators, so they can share the policy or \( Q \)-table. Similarly both elevator agents can share the \( Q \)-table. The system requires only 7852 states in two \( Q \)-tables, whereas the usual implementation would require approximately \( 10^{17} \) states considering only the available information in the system (boolean value signals of all internal elevator buttons, hall call buttons, elevator status, and current floor of the elevators: \( 2^{24} \times 2^{22} \times 3 \times 12^2 \approx 10^{17} \)).

4. Simulation Results

To investigate the performance of the proposed elevator group control system, simulations are carried out approximating the elevator system dynamics by the following parameters: Floor time (the time to move one floor at maximum speed): 3 sec; Stop time (the time needed to decelerate, open and close the door, and accelerate again): 8 sec; Turn time (the time needed for a stopped car to change direction): 1 sec; Load time (the time for one passenger to enter or exit a car): 1 sec; Car capacity: 15 passengers;

Q-learning algorithm and \( \epsilon \)-greedy policy with decreasing value of \( \epsilon \) are used. The initial value of \( \epsilon \) is 0.39 and it decreases at a rate of 3% per day up to the value of 0.03. The discount factor for the calling agent \( \gamma_c \) is set to 0.5, and for elevator agent \( \gamma_e \) is set to 0.95. Since an elevator may reach the calling floor unexpectedly, we consider a comparatively low value of \( \gamma_c \) to reduce the noise effect. The learning rate \( \alpha \) is set to 0.1, and the constants \( a \), \( \lambda \) and \( b \) of eq. (2), eq. (5) and eq. (6) are set to 0.015, 0.9 and 0.1 respectively.

The passenger-arrival data are generated using a typ-
ical hourly passenger-distributions for 24 hours a day considering approximately 2500 incoming passengers per day for the residential building. Figure 4 shows the incoming passengers-distribution for both up-going passengers (passengers who want to go up from their current floor) and down-going passengers (passengers who want to go down from their current floor) per hour. 5% of them are inter-floor passengers (whose starting or ending floor is not the first floor), and the origin/destination of other passengers is the first floor. These typical traffic distribution patterns are approximated from the observed data in a survey on a real residential building having twelve floors and 120 apartments (10 apartments per floor).

To compare the performance of the proposed task-oriented system in controlling the elevator group, we have tested the system using collective RL (CRL) algorithm with omniscient reinforcement and parallel architecture proposed by Crite\(^4\). CRL is considered as the best elevator control method specially for peak down elevator movements in both low and high passengers densities compared to the other existing elevator group control systems. We chose it to compare the various aspects of our method for different traffic profiles of the residential building.

The convergence of TORL system is found much faster than CRL system. The TORL system converges rapidly without any fluctuation in its performance as shown in Fig. 5. The CRL system fluctuates up to approximately 170 days in learning and after that it tends to converge. The learning rate of CRL system is kept at low value to avoid instability in learning as suggested by Crite\(^4\). Also the CRL system has many constraints to force the elevators to stop by the rule, so the decision making points (by learning) are much fewer than TORL system. These cause CRL system much slower in convergence compared to the TORL method specially in lower to medium traffic densities. The TORL system shows shorter waiting time after convergence for the mixed traffic with low to medium density.

Figure 6 shows the comparative performance of both systems in handling different passenger densities after the convergence in learning with the data shown in Fig. 4. The period of each simulation is one hour. So, the number of passengers in this one-hour period defines the passenger density. For each passenger density, the ratio between up- and down-going passengers is chosen randomly within the range between 3/7 to 7/3 to produce 720 data sets with different ratios. The figure shows the averages over the 720 data sets. The TORL system has kept the waiting time shorter compared to the CRL system up to 270 passengers per hour. It became slightly worse at very large traffic. In the task-oriented principle, the elevator is directed only for a specific call, so an elevator does not move to hunt a passenger in other floors (except for parking in the suitable floor) without being called. On the other hand in CRL method, elevators always hunt for the passengers from 1st floor to 12th floor and then the other way round. That is good only if there are always some waiting passengers in some floors, but for lower traffic it gives worse result.

The operating costs in terms of duty ratio of elevators for both systems are shown in Fig. 7. The elevators’ running time is summed up and expressed as a percentage to the total elapsed time. A low duty ratio implies low operating cost. The TORL system requires less elevator movements in both low and high passengers densities compared to the CRL system since the elevator movements are directed by passenger arrival only. The elevators of CRL system always move to hunt the passengers along its up-down axis that makes it costly when the system does not have very high traffic.
Both TORL and CRL methods are also tested for three elevators system and four elevators system for the same building. Both systems have shown shorter waiting time as the number of elevator increases, and still the TORL system has shown better performance up to a certain number of passengers and could keep the elevator movements approximately the same for all cases, whereas for the same traffic, the CRL system has become more costly as the elevator movement increases proportionally with the number of elevators.

To investigate the effect of learning of elevator agents to park the car, the TORL system is tested and compared with the system with no learning of elevator agent, using the same traffic profile for each day as shown in Fig. 4. Figure 8 shows the comparative hourly performance of both systems after learning for a day taking an average value over 30 days, where "C Agents" means that the calling-agents only learn and C&E Agents means the case where both calling agents and elevator agents learn. It is found that the average waiting time at the hours with higher traffic density are long and almost the same for both systems as the both elevators are almost always busy and they do not have time to park the elevator. But at comparatively lower traffic hours (at the hours 2 to 6 and 21 to 24), the TORL system with C&E Agents has shown shorter waiting time compared to the system with no learning for parking the car by the elevator agent. In these low traffic hours they can help to minimize the waiting time by parking the elevator car in suitable floor. This reveals the necessity of learning to park the car in suitable floor at lower traffic hours.

5. Discussions

TORL method offers a number of advantages which cannot be achieved using monolithic reinforcement learning method. First, the task decomposition limits the size of state-spaces hence ensures less computation requirements, which leads the system achieving faster convergence. Second, thinking from the viewpoint of the task leads the agent to choose an action more precisely. This makes the system flexible since the agent gets a gist of the task from state information. So learning process goes well regardless the system dynamics. But in the task-oriented learning method, a human designer is needed to decompose the task into some sub-problems and to define their relations.

The collective RL method can work in satisfactory level as there are many constraints to the agent, such as it must move up or down from the stopped floor, and it can not stop at a floor where other elevators is stopped or there is no waiting passengers. So only the control decision is to stop or continue at the next floor if there is any waiting passenger. These constraints make this system better than other systems for very high traffic as it always moves from the 1st floor to the upmost floor and the way round to hunt for the passengers. But when the traffic is not so high the movements of the elevators worsen the waiting time as well as operating cost. The both methods use multiple agents to control the elevator group system. The task-oriented thinking makes the system flexible (elevator moves only if it is called due to a passenger arrival) for any traffic pattern. On the other hand, the CRL method does not think from the task viewpoint and moves without any specific information of the task, that makes it non-flexible for changing traffic pattern.

6. Conclusions

A physically implementable simple adaptive control method using task-oriented reinforcement learning has been proposed for the elevator group control system, which has shown significantly improved performance compared to the other adaptive algorithm. Faster convergence and simpler implementation of the TORL method recommends it in designing adaptive controller for typical elevator group.

Acknowledgement

This research was partly supported by the 21st Century COE program: Reconstruction of Social Infrastructure Related to Information Science and Electrical Engineering.

(Manuscript received May 20, 2004, revised Feb. 15, 2005)

References

M.A.S. Kamal (Non-member) received the B.Sc. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET) (former Bangladesh Institute of Technology (BIT) Khulna) in 1997. After his graduation he joined the same University as a Lecturer and left his position for higher study in 2000. He received M.S. degree in Information Science from Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan in 2003. Presently, he is pursuing the Ph.D. degree at the same University. His current research interests include reinforcement learning, neural networks, evolutionary algorithm, fuzzy systems and their applications. Mr. Kamal is a member of IEB and a student-member of IEEE.

Junichi Murata (Member) received the Master and Doctor of Engineering degrees from Kyushu University, Japan, in 1983 and 1986, respectively. He then became a Research Associate and an Associate Professor at Graduate School of Information Science and Electrical Engineering, Kyushu University. His current research interests are neural networks, self-organizing systems and their applications to control and identification. Dr. Murata is a member of SICE, ISCIE, IEEE and IEEJ.

Kotaro Hirasawa (Member) received the B.S. degree and the M.S degree in electrical engineering from Kyushu University, in 1964 and 1966, respectively. From 1966 to 1992, he was with Hitachi Ltd., where he served as a Vice President at the Hitachi Research Laboratory. He also served as a Professor from 1992 to 2002 in Kyushu University. He is currently a Professor of Graduate School of Information, Production and Systems, Waseda University. Dr. Hirasawa is a member of IEEE.